# Chapter 8

# Interpretational Semantics

*Summary and Advertisement*

A central insight of the seventeenth century was that mental meaning cannot be understood in terms of resemblance. If the semantic relations between mind and world cannot be understood on the hypothesis that the mind is *like* the world, literally sharing properties with the things it represents, how can it be understood? In the hands of Locke and his successors, covariance replaced resemblance. But whatever advantages this had for Locke, mental representation cannot be understood in terms of covariation by those who want to follow the CTC in supposing that mental representation *explains* how systems manage to get into states that covary with the states of the world. And the attempt to understand mental representation in terms of adaptational roles also appears to reverse the explanatory order central to the CTC, and to be inconsistent with the thesis that cognition (and hence representation) supervenes on abstract formal structure that need not be historically specified.

What is next? The demise of similarity, idealized covariance, and adaptationist theories leaves us with only one candidate: functional role. The approach I will sketch in this chapter—"Interpretational Semantics"—turns out to be ontologically equivalent to a kind of functional-role semantics. However, Interpretational Semantics has, to my intellectual palate, a very different flavor than functional-role semantics; it is motivated in a very different way than typical functional-role theories, it directs our attention to quite different issues, and it generates a

different dialectic. These virtues, I think, swamp the significance of the underlying ontological equivalence I will eventually notice in the next chapter.

Interpretational Semantics is really not an alternative to the traditional theories, which are best construed as answers to a different question. Interpretational Semantics is an account of representation in the CTC, whereas most philosophical discussion of mental representation has to do with Intentionality—i.e., with the contents of thoughts—rather than with the contents of the representations of a computational system. The theory I called the Representational Theory of Intentionality in chapter 2 would forge a tight link between representation and intentionality. But this, as we will see in chapter 10, is extremely problematic if we accept the CTC and the Interpretational Semantics that (I claim) inevitably goes with it. The kind of meaning required by the CTC is, I think, not Intentional Content anymore than entropy is history. There is a connection, of course, but at bottom representation in the CTC is very different from intentionality. To keep this point in the foreground, I will eventually introduce the special term *s-representation* (for simulation-based representation) to stand for the kind of representation described below. The thesis of Interpretational Semantics is that s-representation is the kind of representation the CTC requires to ground its explanatory appeals to representation.

It seems clear that if we are going to understand what representation is in the CTC, we should begin by understanding clearly what it does, and how. Representation is an explanatory construct in the CTC. Until we understand its explanatory role in that framework, we have no serious chance of understanding how it should be grounded for that framework. In order to get a clear picture of how the CTC proposes to exploit representation in the explanation of cognition, it is essential that we also gain a clear picture of the role of computation, for the main thesis of the CTC is that cognitive systems cognize by computing representations. The concepts of representation and computation are correlative in CTC and must be understood together.

*Explaining Addition*

The CTC proposes to explain cognitive capacities by appeal to representation and computation in exactly the way that arithmetical capacities of calculators (such as addition) are standardly explained by appeal to representation and computation. The main explanatory strength of the CTC is that it proposes to explain cognition in terms of antecendently understood notions of representation and computation—notions the CTC takes to be unproblematic in a variety of familiar noncognitive contexts such as calculating and elementary computer programming. We would do well to begin, then, by reviewing the "received" explanation of addition in adding machines.

To add is to be described by the plus function, $+(\langle m, n\rangle) = s$. Hence, to explain what makes a system an adder is to explain the fact that the system is described by +. But + is a function whose arguments and values are numbers; and whatever numbers are, they are not states or processes or events in any physical system. How, then, can a physical system be described by +? How can a physical system traffic in numbers and hence add? The answer, of course, is that *numerals*—i.e., representations of numbers—can be states of a physical system, even if the numbers themselves cannot. A physical system adds by trafficking in numerals and hence, indirectly, in the numbers those numerals represent.

The input to a typical adding machine is a sequence of button pressings: $\langle C, M, +, N, =\rangle$, i.e., $\langle$*clear, first addend, plus, second addend, equals*$\rangle$. The output is a display state, $D$, which is a numeral representing the sum of the two addends. We may think of the button-pressing sequences as arguments to a function $g$ that gives display states as values. An adding machine *satisfies* $g$; that is, the arguments and values of $g$ are literally states of the physical system.[1] Addition, as was remarked above, relates numbers, not physical states of some machine, so a physical system cannot literally satisfy the plus function. What an adding machine does is *instantiate* the plus function. It *instantiates* addition by *satisfying* the function $g$ whose arguments and values represent the arguments and values of the addition function, or in other words, have those arguments and values as interpretations.

Putting all this together, we have that something is an adding machine because

1. we can interpret button pressings (or the internal states they cause) as numbers,[2]

2. we can interpret displays (or the internal states that cause them) as numbers.

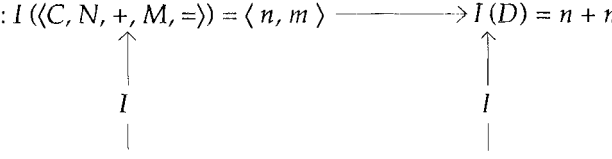
3. the device causally associates sequences of button pressings with displays (i.e., it satisfies a button-pressing-to-display function),

and, given all this,

4. if a token of the button-pressing events interpreted as $n$ and $m$ were to occur, then a token display event interpreted as $n+m$ would normally occur as a consequence. The device instantiates the addition function by satisfying the function mentioned in step 3, for that function is interpretable as the addition function.

To get a physical device to add, you have to get it to satisfy a function interpretable as addition. And that means you have to design it so that getting it to represent a pair of addends causes it to represent their sum.

The following is a useful picture of this whole conception of adding machines. I call it the Tower Bridge picture because it reminds me of London's Tower Bridge.




The top span pictures the function instantiated: +, in our present case. It takes a pair of numbers onto their sum. The bottom span corresponds to the function satisfied (called simply $g$). It takes a

quintuple of button pressings onto a display. The vertical arrows correspond to interpretation: $I(\langle C, N, +, M, = \rangle)$ is the interpretation of $(\langle C, N, +, M, = \rangle)$, namely $\langle n, m \rangle$, the pair of numbers represented by $N$ and $M$. $I(D)$ is the interpretation of the display, which will be the sum of $n$ and $m$ if the thing works right. Under interpretation, the bottom span is revealed as an instantiation of the top span; computation is revealed as addition.

*Computation*

We have reduced the problem of explaining addition to the problem of explaining why the machine satisfies $g$, the function that instantiates addition. In standard adding machines, the values of $g$ are *computed* from the corresponding arguments; adding machines add by computing the appropriate representations.

Of course, functions need not be computed to be satisfied. Set mousetraps satisfy a function from trippings to snappings without computing it, and physical objects of all kinds satisfy mechanical functions without computing them. The planets stay in their orbits without computing them. Missiles, on the other hand, compute their trajectories (sometimes), and, in general, complex systems often satisfy functions by computing them. Humans, in particular, routinely satisfy functions by computing them. This is how recipes and instruction manuals enable you to satisfy functions you don't know how to satisfy directly. The recipe for hollandaise sauce, for example, specifies such things as eggs and double boilers as inputs and hollandaise sauce as output. When you execute such a recipe, you satisfy a function having the recipe inputs as arguments and the hollandaise (the recipe output) as value. Recipes analyze such functions into simple functions you know how to satisfy directly—e.g., add three eggs to the contents of the bowl.[3]

To compute a function $g$ is to execute a program that gives $o$ as its output on input $i$ just in case $g(i) = o$. Computing reduces to program execution, so our problem reduces to explaining what it is to execute a program.

The obvious strategy is to exploit the idea that program execu-

tion involves *steps*, and to treat each elementary step as a function that the executing system simply *satisfies*. To execute a program is to satisfy the steps.

But what guarantees that the steps are executed—i.e., satisfied—"in the right order"? Program execution is surely disciplined step satisfaction. Where does the discipline come in? The discipline takes care of itself. Functions satisfied by $d$ specify causal connections between events in $d$, so if $d$ satisfies $f$ and $g$ and if the current value of $f$ is an argument for $g$, then an execution of the $f$ step will produce an execution of the $g$ step. Program execution reduces to step satisfaction.[4]

You won't go far wrong if you think of it this way. Imagine the program expressed as a flow chart. Each box in the chart represents a step. To execute a step is to satisfy its characteristic function, i.e., the function specified by the input/output properties of the box. If you think of the arrows between the boxes as causal arrows, the result is a causal network with steps (i.e., functions to be satisfied) at the nodes. A system executes the program if that causal network gives the (or a) causal structure of the system.[5]

*The Role of Representation in This Explanation*

The arguments and values of the button-pressing-to-display function that play the role of representations—i.e., the button-pressing sequences and the display states themselves—need not be symbols with any use or meaning outside the system that computes them. What matters is only that the system satisfy a function $g$ that instantiates +. That is, there must exist an interpretation $I$ such that

$$g(x) = y \quad \text{iff} \quad +(I(x)) = I(y).$$

It is, in short, simply the fact that $g$ is isomorphic to + that makes the arguments and values of $g$ representations of numbers for a system that satisfies $g$. It is sufficient for the arguments and values of $g$'s being representations of addends and sums that there exist an interpretation mapping $g$ onto +: The arguments

and values of $g$'s being representations of numbers is *constituted* by the fact that $g$ instantiates +; representation is just a name for the relation induced by the interpretation mapping between the elements of $g$ and the elements of +.

That there is no "further fact"[6] to representation in this case beyond $g$'s instantiating + is obscured by the use of symbols with a meaning that is independent of their use in the system. One is tempted to suppose that the display state that looks like "5" would mean five regardless of whether $g$ actually instantiates +. And so it would—*to us*. But its meaning in the system might be something quite different. To see this, we simply have to imagine that the buttons are mislabeled and that the display wires are crossed. In such a case we would have to discover what represented what, and this we would do by establishing a mapping, $I$, between $g$ and + such that

$$g(x) = y \quad \text{iff} \quad +(I(x)) = I(y).$$

Indeed, it suffices to notice that it *makes sense* to suppose that the buttons *could* be mislabeled, or that the display wires *could* be scrambled, for this would make no sense if the meanings of input and output states were determined by the conventional meanings of those states considered as symbol tokens. We say that a display is scrambled because the conventional meanings of the display don't match the meanings that they have in the system, and this evidently requires that displays have a meaning in the system that is independent of their conventional meanings.

We are now in a position to see that it is somewhat misleading to speak of the explanatory role of representation in this context. The explanatory burden is carried by the fact of simulation and the correlative concept of interpretation. There is a sense in which an adding machine adds because it represents numbers, but there is a more important sense in which it represents numbers because it adds: We can speak of it as representing numbers only because it stimulates + under some interpretation.

From an explanatory point of view, what interpretation provides is a link between mere state crunching (button-pressing-to-

display transitions) and *addition:* Under interpretation, the state transitions of the system are revealed as adding. You can get a feel for the explanatory role of interpretation by imagining that the device is an archaeological discovery. Perhaps you know from documents that the thing is an adding machine; however, looking at it, you have no idea, at first, how to give inputs or read outputs. You can "make it go," let's say, by turning a handle, but this is just more or less random manipulation until you find an interpretation that reveals the state transitions of the device as sum calculations and hence establishes the representational status and contents of those states. To know how to use it is to know how to see it as simulating +, and that means knowing how to "make it go" and how to interpret it.

The concept of representation that is operative here can be brought more clearly into focus by considering again Galileo's ingenious use of geometrical figures to represent mechanical magnitudes (figure 8.1). The crucial point is that, given Galileo's interpretation of the lines and volumes, the laws of Euclidean geometry discipline those representations in a way that mirrors the way the laws of mechanics discipline the represented magnitudes: The geometrical discipline mirrors the natural discipline of the domain. That is, geometrical relationships among the symbols have counterparts in the natural relations among mechanical variables[7] in such a way that computational transformations on the symbols track[8] natural transformations of the system. That is what makes it correct to say that the symbols—lines and volumes—*represent* times, velocities, and distances.

Of course, this is what Galileo *intended* them to represent; that is the interpretation he stipulated. But it is one thing to intend to represent something, another to succeed. Galileo's figures *actually do* represent mechanical variables because the computational discipline *actually does* track the natural one. This is the natural discipline we have in mind when we say that the system behaves according to natural law. I call Galileo's interpretation a proper interpretation because under that interpretation the natural system and the geometrical system that represents it march in step: The geometrical system *simulates* the natural one.[9] A proper

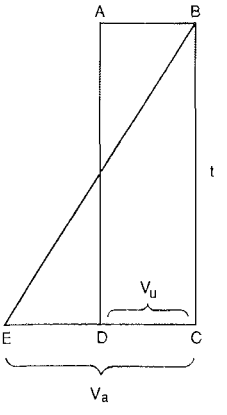
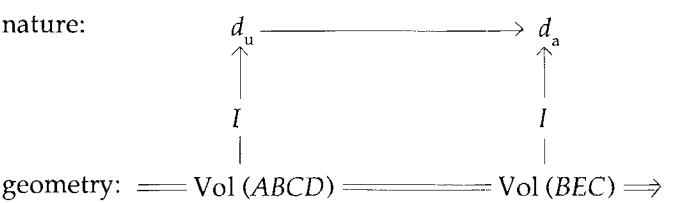



Figure 8.1
Galileo's diagram.

interpretation is an interpretation that gets it right: The symbols actually represent what the interpretation says they represent.

Representation, in this context, is simply a convenient way of talking about an aspect of more or less successful simulation.[10] The volumes behave in the geometrical system in a way analogous to the way certain distances behave in the natural system. Hence, the volumes are said to represent those distances; those distances are proper interpretations of those volumes. For instance, the volume of the triangle tracks the distance traveled by the uniformly accelerated body; the volume of the triangle is the *geometrical analogue* of the distance traveled by the accelerated body. This is what makes it correct to say that the volume of the triangle *represents* the distance traveled by the accelerated body, i.e, that the distance traveled by that body is a proper interpretation of that volume. Representation enters into this story in a way exactly analogous to the way it enters into the story about adding machines. In both cases, it is the fact that one function simulates the other under a fixed interpretation that makes it possible to think of the arguments and values of one function as representing the arguments and values of the other. The causal structure of an adding machine—the fact that it executes an appropriate pro-

gram and hence satisfies the function $g$—guarantees that the arguments and values of $g$ track the numbers; it guarantees, for example, that "3" is the computational analogue (in the machine) of three in the addition function. This is what makes it possible to think of "3" as a symbol *in the system* for three. Analogously, the formal structure of Euclidian geometry guarantees that the volume of the rectangle in Galileo's figure will track the distance traveled by the unaccelerated body, and this is what makes it possible to think of that volume as representing that distance.[11] The analogy between Galileo's geometrical treatment of mechanics and the computational treatment of addition is brought out by noticing that the Tower Bridge picture applies naturally to the former as well as the latter:




Nature maps the distance traveled by the unaccelerated body onto the distance traveled by the accelerated body; geometry maps the volume of the rectangle onto the volume of the triangle.

*s-Representation*

The concept of representation invoked by the CTC is the same concept that is implicit in the sort of mathematical science that Galileo invented. It is the sense in which

- a graph or equation represents a set of data
- a linear equation represents the relation between time to solution and the absolute value of the mutiplier when a product is computed by a successive addition algorithm
- a parabola represents the trajectory of a projectile
- intelligence cannot be represented on a ratio scale (when we say "twice as smart," we misrepresent intelligence),

and

- we ask whether social or economic dynamics can be adequately represented by a set of linear equations (knowing that the nonlinear case is computationally intractable).

It will be useful to give this sort of representation a special name. I will call it "s-representation"—"s" for "simulation," because s-representation is simply a consequence of (one might almost say an artifact of) simulation.

Since s-representation is familiar from the context of mathematical modeling, it is useful to list several features of s-representations that are uncontroversial in that context:

- When we are dealing with mathematical models of a natural phenomenon, the criteria of adequate representation are just the criteria of adequate modeling. Typically, we use a battery of statistical techniques to determine how well a mathematical model simulates nature's discipline observed and recorded as "the data." A failure of fit between data and model shows that the model does not adequately s-represent the world (or that aspect of the world that we are trying to s-represent).
- We draw a strong distinction in this context betweenwhat we intend to represent, or are trying to represent, and what we succeed in representing. No one would make the mistake of thinking that a linear equation represents the relation between time and distance in free fall simply because that is what someone intended or believed.
- s-representation is also explicitly a matter of degree; the issue is *how* adequate a model is and whether it is better than competitors.
- s-representation is relative to a particular target: A particular linear model may be a better model of system $S$ than it is of system $S'$, but it would be out of place—a misunderstanding—to ask which system, if either, it "really" represents.

- Failures of s-representation are often not "localizable": When a model is not adequate, it is sometimes possible to pin the blame on a particular culprit (the wrong empirical value for a parameter, say), but notoriously, it is often not possible to make more than rather vague and global judgments of culpability.
- There is often a pragmatic element to representation in this context: A linear model of a complex social system may be an adequate representation for some purposes and not others, and may be preferable to nonlinear models simply because it is mathematically more tractable.

All of these points (and, no doubt, more that I haven't thought of) apply to representation in a computational context. When we write programs—even very simple ones—we often *intend* interpretations that turn out to be Improper. The computational discipline we have designed is inadequate, and the desired tracking fails, at least in part. Discussing a bridge-playing program, I might say, to myself or to a colleague, "This data structure represents the opponent's bridge hand." But if the program is a failure, so is the representation. The upper and lower spans of the Tower Bridge fail to connect properly. Strictly speaking, what I should say in such a case is "I meant this data structure to represent (s-represent) the opponent's hand, but it doesn't seem to work."[12] This commonplace experience of programmers highlights the fact that successful representation in computational systems is not simply a matter between the symbol and its interpretation but depends essentially on the processes that have computational access to (are "defined over") the symbol. A bug in the "logic" of a program can lead to a failure of representation—a failure that may well be correctable without altering the data structures in question at all. It follows from this consideration that misrepresentation in programming, like misrepresentation in mathematical modeling, may be a global affair, with no obvious culprit. It is often difficult to say whether unsatisfactory performance is best improved by altering the data structures (i.e., by finding a better knowledge representation), or by altering the processes that operate on them.[13]

As was pointed out above in connection with addition, the fact that s-representation is simply an offspring of more or less successful simulation is often obscured by the common use of "near English" (or some other natural language) in coding data structures in high-level programming languages. If a data structure for a program intended to play bridge has symbols such as "K-clubs," it is natural to suppose that what we have is a representation of the king of clubs. And so we do; that is what "K-clubs" means *to us* in "near English." But "K-clubs" doesn't succeed in representing the king of clubs *in the system* if the program is a radical failure. We must be careful to distinguish what we intend a data structure to represent in the system we are building from what, if anything, it *does* represent in the system, and to distinguish both of these from what it represents in some independent representational scheme such as our own natural language. There is no "further fact" required for successful s-representation beyond what is required for successful simulation.

In programming, as in mathematical modeling, failure comes in degrees; the tracking may be imperfect but not fail utterly. The computational and natural systems mostly march in step, but with lapses in coordination. We can bring out this point, and the analogy with mathematical modeling, by imagining Galileo's system "automated," i.e., realized as a computational system for solving problems of mechanics. As Galileo set up the system, it isn't quite right. Moreover, given the geometry he was working with, there *is* no way to get it exactly right. The Galilean geometry misrepresents mechanical reality; it does not, and cannot, perfectly simulate the natural order. Galileo had an inadequate "knowledge representation," but one that went a long way nonetheless. Thus, in the world of s-representation, misrepresentation differs from failure to represent only in degree; failed representation becomes misrepresentation when the failure isn't too bad. This is especially true if the failures are identifiable and (more or less) correctable or avoidable.[14] We usually think of the case of Galileo as one of misrepresentation rather than one of failure to represent, because his system is a great deal better than

the competition was and because we know how to improve performance; that is, we can specify conditions under which the system's performance is quite good—correct "for all practical purposes"—so there is a sense in which we think of it as "being on the right track."

But isn't it more correct to say that Galileo intended to represent mechanical reality but failed? Misrepresentation is representation, after all, not simple failure to represent.

We could say that Galileo failed to represent mechanical reality rather than that he misrepresented it. "Absolutely speaking" (whatever that means, exactly), it is a failure. Relatively speaking—relative to historical context and available competitors—it was a ground-breaking success. Thus, we could say either that Galileo came closer to representing mechanical reality than the competition, or that he represented mechanical reality better than the competition. If we think of matters in the first way, we will not speak of misrepresentation except as a synonym for intended representation that fails to some extent. If we think of matters in the second way, we will say that Galileo misrepresented mechanical reality, but not as badly as, say, Buridan. The important point is that the imagined objection (in quotation marks above) simply presupposes that representation isn't a matter of degree. The underlying thought is something like this: "Either we have representation or we don't have it, and only if we have it can we speak of misrepresentation."

It is all too easy to think of representation as like identification (the speech act), and of the processes that act on them as like predication. Successful identification (hence reference of one kind) is prior to predication, and hence prior to truth. First there is the issue of what, if anything, you have managed to identify; *then* there is the issue of what to predicate of it, and whether the result is something true. But we cannot suppose analogously that first there is the issue of what (if anything) is represented and then there is the issue of what processes act on the representations. What is s-represented is essentially a matter of the processes, for it is essentially a matter of simulation. And simulation is essentially a matter of degree.

s-representation is relative to a target of simulation. Since (as we will see shortly) proper interpretations are not unique, $g$ may be an imperfect simulation of $f$ but a perfect (or better) simulation of $g$. Indeed, given any $f$ imperfectly simulated by $g$, it is (I suppose) always possible to find an $f'$ that $g$ simulates better than $f$. Thus, it makes no sense to speak of s-representation *simpliciter*; s-representation (and, hence mis-s-representation) must be relativized to the function simulated. Given an interpretation function $I$, $g$ is a more or less accurate simulation of $f$. Against this background, we can say that some symbol—one of $g$'s arguments or values—accurately tracks (i.e., represents) or imperfectly tracks (misrepresents) its interpretation under $I$. But the very same symbol may be said to represent or misrepresent something entirely different given a different interpretation under which $g$ simulates (more or less successfully) a different function, $f'$. Galileo's symbols are imperfect representations of mechanical variables, but they are perfect representations of plane figures in Euclidian space. Geometry is better geometry than mechanics. Big surprise.

Once we admit these relativizations, we can see why nothing comparable to the "disjunction problem" that arises in connection with covariance theories arises for Interpretational Semantics. We needn't *worry* that we can always trade misrepresentation of $x$ for accurate representation of Something Else; we *can* do that, but it doesn't *matter*. The fact that a system satisfying $g$ thereby perfectly simulates $f$ under $I$ has no tendency to show that the system doesn't also thereby imperfectly simulate $f'$ under $I'$. And it is the fact of simulation (and its degree of adequacy) that bears the explanatory load in the Tower Bridge strategy. No doubt adding machines simulate functions other than +, but that does not compromise the standard explanation of addition in adding machines.

Is this playing fair? Couldn't the covariance theorist pull the same stunt? Here is how it would go: We prefer imperfect correlation with $C$ to perfect correlation with $B$ on the grounds that $C$, but not $B$, is an important property of $D$, where what we are trying to do is explain the system's performance in $D$ (not $D'$,

where $B$ looms large).

I have some sympathy with this line of defense, but I suspect that this sort of relativization (or lack of uniqueness, if you like) is just what the covariance theorist was trying to avoid. I leave it to the reader to determine whether covariance is worth saving (for some framework other than the CTC) and, if so, whether this move will help.

But notice that Interpretational Semantics allows for cheap representational contents, and learns to live with them by pointing out that their low price doesn't compromise their explanatory role in the CTC. That defense isn't available to the covariance theorist, because covariance theories cannot explain what representation is in a way that is consistent with the CTC.

*Interpretation*

There is a deep problem about the notion of interpretation that needs to be canvassed before we go any farther. The simple way to understand an interpretation function is to think of it as any one-to-one mapping. But this is evidently much too liberal. To see why, notice that there is a one-to-one mapping between multiplication and addition. Thus, if + interprets $g$ (our button-pressing-to-display function), then so does $\times$ (by transitivity of one-to-one mappings). It will follow that $g$ instantiates $\times$ as well as +; i.e., anything that is an adding machine is automatically a multiplication machine as well. This is surely unacceptable.

Given this simple-minded and liberal understanding of interpretation, the whole business of getting a system to instantiate a given function $f$ becomes *trivial*. It is not trivial in general; designing calculators was a major achievement, and designing calculators that multiply as well as add was a nontrivial achievement as well. So there must be something wrong with understanding interpretation simply as one-to-one mappings.

Let us return to our example. Let $A$ be an adding machine that instantiates $+\langle x, y\rangle = z$ as $g(\langle C, N, +, M, =\rangle) = d$, a function that maps button-pressing sequences onto display states. Now, since the multiplication function, $\times(\langle x, y\rangle) = z$, is one-to-one mappable to

+, if we take interpretation to be any one-to-one mapping it will follow that $A$ instantiates $\times$ as $g$ too. But what is the interpretation function? The obvious choice is this:

$$I_{+}(\langle C, N, +, M, =\rangle) = I_{\times}(\langle C, N, +, M, =\rangle),$$

and if $g(\langle C, N, +, M, =\rangle) = d$, then

$$I_{\times}(d) = \times(I_{\times}(\langle C, N, +, M, =\rangle)).$$

There are a number of other possibilities, but—and here are the crucial points—*they all involve computing* $\times$, *and they all involve ignoring the display.* And that's cheating! It is evidently *cheating* to build the function to be instantiated into the interpretation function, because if you do *that* then you will need *another* system (yourself with paper and pencil, typically) to instantiate the function *directly*, i.e., without cheating. In the example, you will need a multiplication machine to compute the values of $I_{\times}$. And if you have one of those, why bother with $A$? It is also cheating to interpret the display in a way that ignores the display. If we try to treat $I_{\times}$ as a function of the display, we will rapidly get a contradiction. When we enter two plus two, the display will read "4". So $I_{\times}$("4") = 4, since that's $2 \times 2$. But when we enter one plus three, the display will read "4" again, but we must interpret it as 3, since $1 \times 3 = 3$. $I_{\times}$ can get around this only by ignoring the display—i.e., by taking the display as a context-sensitive symbol whose interpretation is sensitive *only* to the context.

This particular example can be undercut by requiring that interpretation functions be structure-preserving as well as one-to-one—i.e., by requiring that $f$ and $g$ be isomorphic, which addition and multiplication are not. (In addition, no number plays the role played by zero in multiplication.) But requiring that interpretations establish isomorphism is still much too weak. $x + y$ *is* isomorphic to $2\pi(x + y)$, yet a simple adding machine does not compute $2\pi(x + y)$; it computes $x + y$ and the interpretation has to do the rest (compute $2\pi(z)$).

The obvious moral is that the only really interesting interpre-

tation is what we might call *direct* interpretation. But I must confess that I don't know how to define *directness*. It seems that not building the function to be instantiated into the interpretation function should be a necessary condition, as should not ignoring the arguments and values of the instantiating function. But these are very shaky conditions. First, in any finite device, such as an adding machine, it is possible to get around the first problem—the problem of building the target function into the interpretation—by defining the interpretation function via a (no doubt huge) lookup table. Of course, one would construct such a table by doing the forbidden calculation, but I'm not at all sure how to exploit that fact in formulating a careful definition of what I'm calling *direct* interpretation, because one can obviously *use* such a table without doing the calculations required to build it in the first place. One might counter this move by pointing out, with some justification, that table lookup is a form of computation, and hence that the interpretation does incorporate a computation of the function to be instantiated.

However this may be, some legitimate interpretation surely involves nontrivial calculation, at least for some of us. I am not prepared to disqualify base-3 adding machines just because in order to use them I have to do some translations that, intuitively, are as difficult as adding itself. I suppose someone could get so good at $I_\times$ that it would seem like "reading" the product off $A$'s display rather than like calculating it. Something very much like this surely happens when we learn new notations (e.g., Polish notation in logic) or a new language. More important, one doesn't want the notion of interpretation to be tied in this way to what an interpreter has to do; rather, it should be tied to the complexity (or something) of the proposed interpretation function.[15]

The second problem—ignoring the arguments and values of the instantiating function—can be got around by defining an interpretation that is like $I_\times$ but does take the display into account in some trivial way. For example, define $I'_\times$ as like $I_\times$ except that when the display is "5" the interpretation is that there is no answer. This will force an interpreter to look at the display every

time to check that it isn't "5".

Reflections such as these make *direct interpretation* seem like a rather subjective and relativistic affair. Nevertheless, it appears to me to be absolutely central to the notion of function instantiation (and hence computation), so I'm simply going to assume it, leaving to someone else the task of filling this hole or widening it enough to sink the ship. *Something* must account for the fact that instantiating $f$ isn't enough to instantiate every function isomorphic to $f$.[16] I am inclined to accept a kind of transcendental argument for the solvability of the directness problem: The standard Tower Bridge explanation of addition is *correct*, after all, and it presupposes a nontrivial concept of interpretation; therefore, such a concept of interpretation exists. On the strength of this, when I speak of interpretation I will mean *direct interpretation*, assuming there is such a thing. The intuitive content is, after all, fairly clear: Interpretation must be relatively trivial; the system has to do the work, not the interpretation function.

Even on the assumption of direct interpretation, it might seem that if $A$ s-represents anything at all, then $A$ s-represents any content you like. Let $g$ be a button-pressing-to-display function satisfied by $S$, and let $g$ be interpretable as addition. Let $f'$ be identical to + except that $f'(\langle n, m\rangle) =$ Richard Nixon when $n + m = 5$, and $+(\langle n,$ Richard Nixon$\rangle) = +(\langle n, 5\rangle)$. Then $g$ is interpretable as $f'$—the new intrepretation function is just like the old one except that a display of "5" represents Richard Nixon instead of 5. Thus, we seem to have the consequence that representational contents are not just cheap, they are free. But isn't it clear that calculators represent numbers *all the time*, and that they don't represent numbers most of the time and Richard Nixon on occasion?

The trouble with the argument that saddles us with adding machines s-representing Nixon is that, whereas $g$ simulates $f'$, it isn't really the case that "5" is the computational analogue of Richard Nixon. "5" doesn't track Richard Nixon as he is buffeted about by any natural discipline except the discipline enforced on him by $f'$, which is evidently "cooked up." Once again, I am embarrassed by the fact that I have no general account of what makes $f'$ a degenerate target for simulation, but it seems intui-

tively clear that *f′ isn't* a proper object of simulation, and hence that adding machines don't represent Richard Nixon in virtue of simulating *f′*.

Whatever one thinks about degenerate simulation targets like the one just described, it is clear that any system that simulates *f* is bound to simulate a lot of other functions as well—e.g., the numerical function that relates the Gödel number of a standard expression of an argument of *f* to the Gödel number of a standard expression of *f*'s corresponding value. These won't typically be *familiar* functions, or *interesting* functions, but they will be simulated nonetheless. And this still leaves s-representational content relatively cheap, on the assumption that price varies inversely with supply.[17] We have already seen that the availability of alternative interpretations in no way undermines the explanatory use to which s-representation is put by the CTC. Nevertheless, it is tempting to suppose that some interpretations are improper on grounds external to the facts of simulation—viz., on the grounds that only some simulations matter (i.e., are actually used by a containing system). The idea here is that an s-representation of *f* isn't a representation of *f to the system itself* unless s-representing *f* actually has a function in the system. Both Millikan and Dretske utilize this idea, though in different ways. The basic idea is to choose, from among possible interpretations, the interpretation that tracks the function of the process interpreted, and call *that* the representational content. We might call this the selection-by-function approach to representation—a kind of two-phase approach in which we add a functional requirement to s-representation in order to get "the real thing."

The selection-by-function approach can be made very attractive by the following reflection: Whether the inputs and outputs of a program (I include all the data structures a program constructs as among its outputs) should be considered representations at all, or whether (like eggs and hollandaise) they should be construed simply as inputs and outputs, seems to depend on how they are used—i.e., on their *functions*. The inputs and outputs of a recipe *could* be treated as symbols. There is, after all, nothing intrinsic about eggs and hollandaise that unsuits them for a

representational career; it is just that no one and no thing actually *does* treat them as representations of something else. And this makes it seem that whether something actually is a representation, as opposed to merely being capable of representing, is a matter of whether it actually gets used as a representation.

The trouble with the selection-by-function approach is that it actually adds nothing new; it enforces no new constraint beyond what is already present in s-representation. What is it, after all, for a computational system to use a simulation of $f$? Well, the values of $f$ must be arguments for $h$. Hence, the values of $g_f$ must be arguments for $g_h$. So what we have here is a super-function $F$ having $f$ and $h$ as components, instantiated as $G$, having $g_f$ and $g_h$ as components. In short, we have a computation of the values of $G$. The problem of how the simulation of $f$ is used resolves into the problem of how to interpret the computation of $G$—i.e., into the problem of what the computation of $G$ simulates. The selection-by-function approach provides no constraints beyond what is provided by the requirement that an interpretation of $G$ must interpret its computation, i.e., its component functions, i.e., the steps of the program execution of which explains satisfaction of $G$. This is not a trivial constraint by any means, but it will not rule out the alternative numerical (and other) interpretations whose existence is guaranteed by model theory.

Perhaps we could understand use noncomputationally, as Millikan does. But this appears to be no help either. If $I$ and $I'$ are alternative interpretations of a system $S$ having the same domain and isomorphic ranges, then the physical structure that we specify when we say that $S$ has the s-representational content provided by $I$ is the same physical structure that we specify when we say that $S$ has the content provided by $I'$. For $S$, having $I$-content will be nomologically equivalent to having $I'$-content. Hence, any causal or evolutionary interactions tracked by the fact that $S$ has $I$-content will be tracked equally by the fact that $S$ has $I'$-content. The problem is that any causal/selective interactions will be sensitive only to the physical structure, and we pick out by reference to the one content the very same physical structure that we pick out by reference to the other. So it looks as if noncompu-

tational but nonintentional interactions won't distinguish among alternative interpretations. To put the point intuitively, if somewhat misleading: A robot that "thinks" it is simulating a numerical function $N$ will do just as well in $E$ as one that "thinks" it is simulating natural processes in $E'$, as specified by $f_{E'}$, provided $N$ and $f_E$ are isomorphic. Both will be equally good (or bad) at picking up trash, for example. Narrow content can be a lot narrower than one might have thought!

*Cognition*

The CTC proposes to understand cognitive systems as systems that computationally instantiate cognitive functions. This idea requires a bit of unpacking.

The CTC embraces a thoroughly rationalist conception of cognition. A system is cognitive, according to this conception, in virtue of respecting epistemological constraints appropriate to its task domain (the domain it is said to cognize). That is, its behavior is cogent, or warranted, or rational relative to its inputs and internal states. This is what makes it correct to think of it as cognizing a domain rather than merely responding to an environment. It follows from this conception that we have no reason to think of a system as cognitive except insofar as we can describe what it does—its capacities—in semantic terms, for epistemological constraints are defined only for propositions, or things with propositional contents—things with truth values, in short.[18] We thus arrive at the idea that having a cognitive capacity is instantiating a function that relates propositional contents, i.e., a function that takes propositional contents as arguments and values and relates them as premises to conclusion. A cognitive system, in short, is an *inference engine*—a system that merits an inferential characterization. Thus, to explain cognition is to explain how a system can merit an inferential characterization—i.e., to explain how it can reason. This is the conception I put forward in Cummins 1983. It is explicit also in Haugeland 1978, Dennett 1978, 1987, and Pollock 1987a. It is at least implicit in the work of those (e.g., Fodor) who think of cognitive systems as systems that realize some form of belief-desire psychology; as

Dennett points out, belief-desire psychology is essentially a psychology of rationality. Under this conception, the problem of cognition becomes the problem of explaining the fact that the system is described by a *cognitive function*, or, for AI, of building a system that is described by a cognitive function.[19]

It is possible to think of cognitive functions as relating intentional states (e.g., beliefs and desires) to one another, since epistemological constraints make sense when applied to these as well. If we go this route, we are free to think of physical systems as actually satisfying (rather than instantiating) cognitive functions. But this wiggle really makes no difference. Cognitive science generally assumes that cognition is a matter of generating the right representations. One then supposes either that the representations have as interpretations intensions (the proposition that Nixon is a crook, say) that are the arguments and values of cognitive functions—in which case we think of those functions as instantiated—or that the representations figure as constituents of intentional states, the intentional states inheriting their intentional properties (being a belief that Nixon is a crook, say) form the semantic properties of the representations (the proposition that Nixon is a crook). Either way, the representations wind up having the same interpretations, and the system winds up instantiating the same abstract function.

The CTC's proposal is that cognitive systems are computational systems: A cognitive system is described by a cognitive function because it computes representations whose contents are the values of the cognitive function, and computes these from representations of the function's arguments. If we suppose that the objects of computation—the things over which the computations are defined—are *symbols*, some of which are *content-equivalent* to the inferential descriptions that figure in the specification of the target cognitive function, we can explain why the system merits inferential characterization, i.e., why it is cognitive: The system merits inferential characterization because it computes symbols representing conclusions from symbols representing the corresponding premises. To move in a disciplined manner from symbol to symbol—to *compute* symbols—is, *under interpre-*

*tation*, to move in a disciplined manner from content to content. If we get the discipline right, we get inference and hence cognition.[20]

To say that there are cognitive engines, then, is, on this conception, to say that the behavior of certain things is describable via an interpretation that reveals their activity as epistemologically constrained—as rational. Once the rationality of the activity is revealed, we are led to ask how such activity is possible. The strategy employed by the CTC is designed to answer this question by providing a program (computation) and interpretation such that the system executes the program and, under interpretation, execution of the program is revealed as the very cognizing identified as the explanandum. The basic assumption of the CTC is that, under proper interpretation, symbol crunching is cognition.

Semantics enters into the picture because to understand how a physical device could satisfy a cognitive description we need to forge a conceptual connection between computations and inferences (broadly construed). The CTC explains a cognitive capacity by showing how it is computationally instantiated, and this requires linking the arguments and values of cognitive functions. Interpretations specify the links you need to see in order to see computation as cognition (or as addition, for that matter). Objects of computation seen under interpretation—i.e., individuated *semantically*—are representations. We call an object of computation a representation when it is important to see the computation as an instantiation of something else. That is why the display states of a calculator are properly regarded as representations but the outputs of a recipe (as in hollandaise) are not.

It is useful to put the explanatory role of interpretation into a broader perspective. The goal of science is, in part, to develop concepts (or a vocabulary, if you like) that enables us to solve particular explanatory problems. To a first approximation, what a good conceptual scheme does is force us to characterize the system under study in a way that screens out all the information that is irrelevant to the problem at hand, leaving only what is essential. Think of the conceptual scheme provided by Newto-

nian mechanics as a kind of filter, like a pair of conceptual glasses that allow you to see only what matters for the solution of mechanical problems. Provided with a pair of Newton glasses, when you look at a pool table, what you see is a lot of arrows on a plane normal to gravity. The point at which an arrow originates represents the center of gravity of a ball. The length and direction of the arrow s-represent the momentum of the ball. And it turns out that when you see pool tables that way, you can see them simply as conservers of momentum and kinetic energy. Newton glasses, in short, reveal pool as a certain kind of conservation system. Analogously, the proper semantics—i.e., the right interpretation—will allow you to see certain computations as adding and (if the CTC is right) others as cognizing. A proper interpretation of an adding machine takes a physical process and filters out everything but what matters to seeing addition. The CTC is just the thesis that what works for addition will work for cognition.

*The Specification Problem*

I will conclude this chapter by briefly sketching a potential difficulty with the CTC. My goal here is not serious criticism. My point is that the criticism in question gives us another perspective on the CTC's explanatory strategy and, hence, on its use of the concept of representation.

Crucial to the CTC's conception of cognitive theory is the idea that there are independently specifiable cognitive functions. The idea is that cognitive capacities can be specified as functions on contents in a way that is independent of the way any particular system instantiates or computes those functions. The analogy with calculating is exact and historically important. Multiplication, addition, sine, square root, etc. can be specified independent of any instantiating device. The function so specified is the upper span of the Tower Bridge. We can then ask whether there is a lower span (a computation) and towers (an interpretation) that will support that upper span. If the same strategy is going to work for cognition, we will need comparable independent specifications of cognitive capacities to form the upper span.[21]

It is becoming increasingly clear that the specification problem is the hardest part of the program. The CTC—which is, necessarily, rationalist to the core—is just the idea that cognizing generally is like doing science: *Automate* Galileo's geometry and what you have, under the appropriate interpretation, is a system that simulates mechanics. But since this amounts to mapping mechanical propositions onto other mechanical propositions that "follow," it is also cognition. Cognition is just computational simulation of "natural functions" via the instantiation of cognitive functions that give a theory of the natural function in question.

But suppose there is no natural function to simulate. Perhaps remembering faces and finding your way home isn't like doing mechanics—can't be like doing mechanics—because these domains, unlike mechanics, aren't governed by laws special to them. The analogy with science makes this seem rather plausible. After all, science works only where there are natural kinds. A science of temperature *per se* is possible only if, as we say, thermal phenomena form an autonomous domain of inquiry. What this means is that thermodynamics is possible only to the extent that thermal phenomena are governed by laws that describe those phenomena "in their own terms"—i.e., autonomously. In contrast, an autonomous special science of clothing is hardly possible, as Fodor (1984a) has emphasized in another connection, because there are no special laws of clothing. Clothing, of course, falls under laws (e.g., the laws of physics, chemistry, and economics), so there is no question that scientific study of clothing is possible. There are also, no doubt, "rules" in the clothing domain: how to dress for different occasions and effects, what needs ironing, what can be washed, and so on. But it is doubtful that these rules could be compiled into an expert system that captures human intelligence in the clothing domain. Perhaps, as Dreyfus and Dreyfus (1985) argue, expert human performance in such domains is, at bottom, not rule-driven but example-driven. But this would mean that we cannot specify the explanandum as a set of cognitive functions that take as arguments and values states of, or propositions about, "the clothing domain." We can

have a Rationalist specification of cognition only where we can have a special science. But, while we can perhaps have science anywhere, we surely cannot have a special science for every possible domain of intelligent thought.

It seems to follow that the CTC, like mathematical science, will work only for the cognition of autonomously law-governed domains. The precondition for success is the same in both cases: There must be a well-defined upper span to the Tower Bridge. Special science isn't always possible. If cognition is possible where special science is not—in the cases of clothing and faces of conspecifics, for example—then the CTC's Tower Bridge picture of cognition can't be the whole story.[22]

If, as seems increasingly likely, the specification problem for cognition should prove to be intractable, or to be tractable only in special ways, where will that leave us? I think it will leave us with a kind of biological chauvinism (Block 1978). Cognition will simply be identified ostensively, and hence extrinsically, as *what humans do when they solve problems, find their way home, etc.* We will be left, in short, with Turing's (1950) conception that a cognitive system is just one that is indistinguishable from humans (or other *known cognizers*) in the relevant respects.

Perhaps the CTC could learn to live with this eventuality, but not very comfortably, I suspect. It is difficult to see what methodology will move you from function satisfaction (program execution) to Turing equivalence if the *target*—the human, or whatever—has no known or knowable characteristic function that might be analyzed ultimately into steps that have known computational instantiations. Synthesis works best if the thing you are trying to synthesize has a known analysis. Otherwise, it is trial and error. That is *nature's* way, of course, but it takes a long time. For now, anyway, the CTC has no way to make progress without functionally specifying the target and trying for an analysis that will eventually lead to computational instantiations.

# Chapter 9
# Functional Roles

## *Functionalism about Mental Meaning*

Functional-role semantics, applied to the problem of mental representation, is just functionalism applied to mental representations rather than to mental states and events generally. Mental states are individuated functionally, according to functionalism. But some mental states—representations, for example—are individuated by their contents. Hence, functionalism implies that mental contents are individuated functionally.

The two most popular flavors of functionalism are computational functionalism and causal functionalism, and this carries over to functional-role theories of mental representation. According to causal versions of the theory, a mental state (or event, or process, or whatever) is a representation, and has the content it has, in virtue of its causal role. Thus, something in the mind is a |cats eat mice|, say, because of the position it occupies in a causal network. We specify the position of a representation in a causal network by specifying the possible causal paths to its occurrence in the system, as well as the possible causal consequences of its occurrence. Computational versions are exactly analogous, with computational role substituted for causal role. Although functional-role theories almost always look to either computational roles or causal roles, it is useful to be able to discuss a generic theory, leaving it open how functional roles are to be understood. It is in this spirit that I use the term "functional role."

*"Long-Armed" and "Short-Armed" Roles*

Functional-role theories come in single-factor and two-factor forms. The idea behind two-factor versions is that there are two components or aspects to meaning: a functional role, which is entirely "in the head," and an external component, which relates the component in the head to the world (Field 1977, 1978; McGinn 1982; Loar 1982; Block 1986, 1987). Single-factor theories sometimes take functional role to include relations to factors "outside the head" (Harman 1982). For example, a single-factor theory may attempt to incorporate the insights of the so-called causal theory of names by holding that the causal chain that (according to such theories) links the use of a name to an "initial baptism" is part of the functional role of whatever mental representation underlies or is analogous to a name (if any).

Block (1986, 1987) argues that single-factor and two-factor versions are essentially equivalent on the ground that two-factor theories simply divide functional roles into two components: an internal or "short-armed" component and an external component which, when added to the "short-armed" component, makes a single "long-armed" component and hence a single factor.[1] The point of distinguishing an internal and an external factor is just to capture the notorious distinction between wide and narrow content (Putnam 1975; Burge 1979, 1986). All two-factor theories assume that it is possible to factor things into internal and external components—an assumption made plausible by the following reflection: We want to develop a theory of cognition that allows for the possibility of the same cognitive system's encountering radically different environments. The possibility of a two-factor approach follows from this desideratum and from the assumption that the desired theory will still be a content-based theory. The content that is alleged to remain constant across changes in environment is narrow content.

*Computational and Causal Roles*

It is seldom noticed that causal roles are bound to differ from computational roles. This follows immediately from the fact that different causal architectures can instantiate the same computational architecture. But it follows also from the more mundane

point that a given computational state of a system is bound to stand in a mass of causal relations that have nothing to do with its computational status. Only the causal arrows that eventually link the state to other computational states matter.[2] This may, of course, implicate states and processes that themselves have no computational interpretations. But many causal relatives will be left out—e.g., those that are too small, or too distant in space or time, to effect the outcome of the computation. The difference will be especially pronounced if we assume that functional roles are long-armed, for, although there are surely some computational roles that don't stop at the skin, the external causes and effects of representational states will surely have the external computational causes and effects as a small subset. One might try to make the two equivalent by restricting attention to the causes and effects a state has *as a representational state*—i.e., to the causes and effects that a state has in virtue of its representational status and content. But this will not do; the functional-role theorist is trying to define representational status and content in terms of functional roles, so it would be circular to *identify* the relevant functional roles by reference to a state's representational status and content.

Since there are bound to be causal differences where there are no computational differences, but not vice versa, it follows that a causal-role theory allows for distinctions of content that are not allowed by a computational-role theory. This might be thought to show that causal-role theories are inconsistent with the CTC on the ground that the CTC assumes that cognition, and hence content, supervenes on computation. But the CTC needn't assume that every distinction of content derives from a distinction in computational role; it need only assume that distinctions of content not mirrored by computational differences make no difference to cognitive capacities (i.e., to target explananda). An advocate of the CTC is free to accept a causal-role theory provided he or she also holds that, as a matter of fact, distinctions of content relevant to cognition will turn out to be "narrower" (i.e., less finely distinguished) than content in all its causally determined subtlety. It is not legitimate to complain against this that

differences in content not reflected in computational role entail differences in cognition not reflected in computational role and hence entail that the CTC is false. The trouble with this complaint is that it simply *assumes* that every difference in content must make a cognitive difference.

The argument for this assumption is typically that my Twin Earth duplicate or Burgean counterpart will have different beliefs and desires than I, yet these differences will not be reflected in different computational roles. This is true. But it doesn't follow that I and my Twin Earth duplicate or Burgean counterpart have different cognitive capacities in the sense of "cognitive capacity" in which the CTC seeks to explain such things. Just as content may be wide or narrow, so may cognition. The CTC evidently seeks to explain cognitive capacities narrowly construed, i.e., construed as the sort of ahistorical capacities that a system might bring to bear on radically different environments, and that might be realized in radically different stuff.[3]

*Functional Roles and Conceptual Roles*

Though they are often not distinguished in the literature, what I am calling functional roles must be distinguished from what are commonly called conceptual roles. As the name implies, conceptual-role semantics, applied to mental representations, is the idea that a mental state is a representation, and has the particular content it does, in virtue of the role it plays in cognitive processes such as inference. According to this theory, one can locate a mental representation—a |cat|, say—in a cognitive network by considering the possible *cognitive* consequences of occurrences of a |cat| in the system and the possible *cognitive* paths to occurrences of a |cat| in the system.

Conceptual-role semantics, then, is the doctrine that something is a representation, and has the content it does, in virtue of its cognitive role. It follows that conceptual-role semanticists make no provision for representation in noncognitive systems. They thus imply (suggest?) that the representation of numbers in a cognitive system is a fundamentally different affair than the representation of numbers in calculators.[4] Tying representation

essentially to cognition in this way seems perverse from the viewpoint of the CTC, which, as we saw, takes cognitive representation to be of a piece with representation in computational systems generally: It is an absolutely central thesis of the CTC that representation in cognitive systems is exactly the same thing as representation in computational systems generally. This is why computation and representation can be thought of as independently available to *explain* cognition. The CTC's proposal is that cognition is just very sexy computation and representation—phenomena that are uncontroversial and relatively well understood in the context of many noncognitive phenomena. This proposal loses most of its bite if we think of representation as a kind of by-product of cognition. It is rather natural to think of representation in this way if you begin by thinking of representation in terms of belief and desire—i.e., in terms of the propositional attitudes; for then representational systems are the things with beliefs and desires, and these are, as Dennett and many others have pointed out, essentially cognitive systems: Belief-desire explanations make sense only in a context in which cogency conditions are respected. The upshot is that we get representation and content tied essentially to rationality, as in Dennett. In chapter 10 I will argue that it is a mistake to think of representation as a kind of corollary of the presence of beliefs and desires. For now, it is enough to notice that the CTC certainly does not so conceive it: Calculators represent but are not belief-desire systems.

Another thing to notice about conceptual-role semantics is that it provides for narrower contents than functional-role semantics. Functional roles—i.e., causal or computational roles—will allow for distinctions of content not allowed by conceptual roles, since there are bound to be computational and causal links not mirrored by cognitive links. For example, almost all computational processes have "side effects" that are (or should be) irrelevant to the logic of the program. Are we to simply assume that genuine cognitive systems are free from "side effects?"[5] More subtly, part of the computational role of |cat| might include the computational consequences of storing it "at the wrong address," or those of buggy access.[6]

Why allow for distinctions in content that are not reflected in any difference in cognitive role? The answer, of course, is that cognitively equivalent states may yet have different contents (Putnam 1975; Burge 1979). If we can argue that computational or causal factors not mirrored in cognitive role sometimes determine content, then an advocate of the CTC who holds a conception of cognitive capacities that provides for sameness of capacity across different computational or causal realizations will want to distinguish a narrow aspect of content that exhibits the same constancies. But the CTC certainly does not dictate such a move. The CTC must hold that the capacities it seeks to explicate retain their identity across differences in *noncomputational factors*. It must therefore cleave to the viability of a kind or aspect of content that is narrow with respect to causal and historical factors not mirrored in computational architecture. But computationalists are free to choose whether to allow for contents distinguished on computational but noncognitive grounds.

Again, one must be careful with "noncognitive." One must resist the move to *define* cognitive differences in terms of content differences, thus making cognitively equivalent states with different contents a contradiction in terms. From the perspective of the CTC, the distinction is unproblematic; the thing to keep your eye on is that there can be computational processes that are insensitive to differences in meaning that are determined by factors irrelevant to the success or failure of simulation. The computationalist is free to concede that these are differences in content, but not in the kind or aspect of content that matters to the specification or explanation of the capacities targeted by the theory (*viz.*, capacities that remain constant across different environments and histories).

There is a temptation to think that conceptual-role semantics is not even a candidate for a *naturalistic* account of mental representation, since it helps itself to the concept of a cognitive process. Students standardly make this complaint. One then smiles knowingly and *explains*. The standard functionalist strategy, in this case, is to define all cognitive processes and states together in terms of the abstract structure of nonsemantic and noninten-

tional relations (e.g., causal or computational relations) that these processes and states have to one another and to inputs and outputs (see Block 1978). Thus, *ontologically*, the suggestion is the same as in the case of functional-role theories (and interpretational semantics, for that matter): To harbor a representation is to harbor something with the right computational or causal role.

But I think the standard student complaint is on to something. The canonical second-order reduction gives us an ontologically naturalistic specification of what it is to be, e.g., a |cat|, but it doesn't provide us with a solution to the problem of representation as we have been conceiving that problem. Conceptual-role theory tells us in what structure of natural relations a state must be imbedded to have a content, thus satisfying the requirements of ontological naturalism. But it doesn't tell us anything about why the natural structure picked out is the right structure beyond telling us this: It is the structure that, in some particular physical system, happens to realize that system's cognitive structure. Thus, when we ask what it is about some particular node in the structure that makes it a |cat|, the only possible answer is this: It is that thing that stands in the computational/causal relations that realize the inferential and other cognitive relations characteristic of |cat|s. And this means that the theory can't give a noncircular answer to the central question about representation.

I do not mean to suggest that the theory can't define what it is to be a |cat| in purely nonintentional, noncognitive terms; it can. The point is, rather, that the theory gives us no clue as to what it is about the abstract structure it picks out, and what it is about some particular node in that structure, that makes *that* structure a cognitive structure, and *that* node a |cat|. It just gives us an ontologically naturalistic identification of the |cat|s without giving us a solution to the problem of representation. It leaves us wondering why occupying a particular position in a network of abstract relations should amount to *having a meaning*.

Notice how all this differs from interpretational semantics. There, we begin with some function $f$ to simulate. Assume that cats appear as arguments or values of $f$. If $S$ satisfies $g$, and $g$ simulates $f$ under interpretation $I$, then the element of $g$ (hence

state of $S$) that $I$ maps onto cats is a |cat| in $S$. The crucial points are these:

> (i) $f$'s being a *cognitive* function is irrelevant to whether something in the system is a |cat|. The interpretational semanticist doesn't tie representation to cognition, and therefore allows for the central insight of the CTC that representation in cognitive computational systems is the same as representation in computational systems generally.
>
> (ii) Interpretational semantics brings out the central fact that it is a state's role in a *simulation* that makes it a representation, and the fact that what makes it a |cat| is that its role in that simulation is to track cats.

Ontologically, conceptual-role semantics and interpretational semantics are equivalent. If a certain abstract cognitive structure $C$ is realized in a computational architecture $A$, then $A$ will simulate $C$, and hence $A$ will have $C$ as a proper interpretation, with the item realizing the |cat| role interpreted as a representation of cats. And if $A$ simulates a cognitive structure $C$, then it certainly "realizes" $C$ in the sense required by the standard second-order reduction. But interpretational semantics provides an explication of the concept of representation (s-representation) and its explanatory role in the CTC that is present in the conceptual role-approach only in a question-begging form.

This equivalence between interpretational and conceptual-role approaches holds whether we take conceptual roles to be long-armed or short-armed, and this has an interesting consequence. If, following Harman, we think of conceptual roles as "long-armed"—i.e., as stretching out into the world, and into the past (and the future?)—then we are also thinking of cognitive architecture and its instantiating architecture as "long-armed." To some this may seem good news, for to see matters in this way is to see cognition as instantiated by a system that includes the environment of the organism. On the other hand, it should give pause to those who looked to organism-environment interactions to make contents unique in a way that they admittedly

cannot be if we restrict ourselves to "what is in the head," for cognitive interpretations of organism-plus-environment systems will be no more unique than cognitive interpretations of "what is in the head." The added complexity, of course, will make viable interpretations that much harder to come by, but there will still be countably many if there are any. It follows from the equivalence between interpretational semantics and conceptual-role semantics that long-armed conceptual roles provide no stronger constraints on representational content than interpretability.

That including the environment doesn't give us interpretational uniqueness should come as no surprise. The idea that relying on connections to the environment could uniquely fix mental content evidently presupposes that there is a unique best way to specify those environmental factors. But what will this be? Since we are specifying roles, our specifications had better be psychologically projectable (because we want regular counterfactual supporting connections with mental states) and physically, chemically, biologically, economically, . . . projectable (because we want—or anyway, are stuck by the theory with—regular counterfactual supporting connections with other environmental events). The idea of "long-armed" functional roles thus presupposes a homogeneously specifiable causal or computational network that covers both organism and environment. Of course there are such networks (the one given by thermodynamics, for example), but the idea that there is a sort of unique causal network here, with the organism being simply a proper part, is simply outlandish. The most plausible candidates—mechanics and thermodynamics—will not serve up the right contents for things like |cat|s and |shirt|s. (See Fodor 1984b.)

The point I made against conceptual-role approaches—that they provide no explication of the concept of representation or of its explanatory role in the CTC—can be made against functional-role approaches: They provide us with no hint as to why being a node in a network of computational or causal relations should make something a representation or endow it with a particular content. Moreover, it must be confessed that what I have been

calling functional-role theories are really straw men; no one actually holds them. A functional-role theorist has no motivation for picking some causal or computational architecture and electing it to the office of representation embedder. The whole idea makes sense only if we suppose that we have some antecedently specified function the realization of which will be a representational system. This, of course, is how interpretational semantics sees matters. From this perspective, the functional-role approach just *is* interpretational semantics with about half the story missing. It is like conceptual-role semantics in not providing an explication of the concept of representation, but it is like interpretational semantics in not tying representation essentially to cognition.

*Conclusion*

The conceptual-role or functional-role approach to representation is a natural by-product of a familiar "quick and dirty" argument for functionalism about mental content. One begins with an ontological question: What is it in a person's head—my head, say—in virtue of which my states are contentful? On what does content supervene? The answer provided by the CTC is relatively straightforward: I am a thinking system in virtue of the computational structure of my brain. Thoughts, and contentful states generally, must be what they are in virtue of their place in that structure. But a thought is what it is in virtue of its content. So having a content—e.g., being a |the cat is on the mat|—must be a matter of occupying the right role in a computational structure! Right?

The trouble with this way of proceeding is that one *begins* with a human being—something assumed to be a thinking system from the start—and *then* asks simply how content must be realized in humans. This opens the way to saying that it is the computational structure of the human brain, whatever that turns out to be, that must do the job. One skips right over the question of what sort of functional structure will realize contentful states, and why. It's just whatever functional structure humans turn out

to have, this being guaranteed to be the right sort of structure (given computationalism) by the fact that humans are known to be cognitive/representational systems.

But functionalism as a theory of mental representation gives us no clue as to why having a computational role should make something a representation or give it a particular content. Indeed, it obscures why computational theories of cognition should traffic in semantics at all. The functional-role formulation provides no hint as to how computationalists should answer Stich's (1983) Challenge: Given that the causal and the computational impact of a data structure are tracked by its nonsemantic properties, their semantic properties need not (and hence should not) enter into any causal or computational explanations that feature them. So why treat them as representations at all?

Functionalism, by identifying mental states with their roles in the production of behavior, suggests that the explanatory project of the CTC is to explain behavior—indeed, to explain stimulus-to-response connections (now called I/O behavior)—at a level of abstraction that isn't neuro-chauvinist by tracking the causal routes through the system via the abstract computational properties of its states. Functionalism thus encourages us to think of mental states as causal mediators, and hence as things whose explanatory role is to make possible a story about the causation of behavior. When a functionalist comes to think of representation, therefore, it is inevitable that the problem should be posed thus: What is the role of representation in the causal mediation of behavior? Once we combine this picture with computationalism, it is hard to see how the concept of representation could do any serious work, and we wind up with Stich's syntactic theory of the mind. Functionalism, since it misrepresents the explanatory role of representation in the CTC, leaves us with Leibniz's worry (see *Monadology*, p. 17). It leads us to ask "What makes a mere syntactic engine a semantic engine?"—a question bound to make us wonder, with Leibniz, how meaning gets into or attached to the cogs and wheels. And once we have come this far, the honest answer is the one Stich gives: Meaning doesn't do any work, so forget it.

Standard over-the counter functionalism therefore leads us to misconceive the use to which the concept of representation is put by the CTC. For the CTC, representation is simply an aspect of simulation. In the special case of cognitive functions, this takes on a special significance for philosophy, though it is nothing special to the CTC, because when the simulated function is cognitive, meaning maps a mechanical process onto a rational one, revealing mind where there was only mechanism. But the concept of meaning has a life in the CTC only as the child of interpretation. Meanings *just are* what interpretations provide. Interpretation is the fundamental notion here; *meaning* is just a name for the product. And to understand the notion of interpretation in the context of the CTC you must understand its explanatory role, which is to map a computational process onto another process, revealing the former as an instantiation or a simulation of the latter.[7]