Cognition, 6 (1978) 229–247

Discussions

Tom Swift and his procedural grandmother*

J. A. FODOR
Massachusetts Institute of Technology

1. Introduction

Rumor has it that, in semantics, AI is where the action is. We hear not only that computational (hereafter 'procedural') semantics offers an alternative to the classical semantics of truth, reference and modality[1], but that it provides what its predecessor so notably lacked: clear implications for psychological models of the speaker/hearer. Procedural semantics is said to be 'the psychologist's' theory of meaning, just as classical semantics was 'the logician's'. What's bruited in the by-ways is thus nothing less than a synthesis of the theory of meaning with the theory of mind. Glad tidings these, and widely credited.

But, alas, unreliable. I shall argue that, soberly considered:

(a) The computer models provide no semantic theory *at all*, if what you mean by a semantic theory is an account of the relation between language and the world. In particular, procedural semantics doesn't supplant classical semantics, it merely begs the questions that classical semanticists set out to answer. The begging of these questions is, of course, quite inadvertent; we shall consider at length how it comes about.

(b) Procedural semantics does provide a theory about what it is to know the meaning of a word. But it's not a brave new theory. On the contrary, it's just an archaic and wildly implausible form of verificationism. Since practically nobody except procedural semanticists takes verificationism seriously any more, it will be of some interest to trace the sources of their adherence to the doctrine.

(c) It's the verificationism which connects the procedural theory of language to the procedural theory of perception. The consequence is a view of the relation between language and mind which is not significantly different from that of Locke or Hume. The recidivism of PS theorizing is among the most striking of the ironies now to be explored.

---

*Requests for reprints should be addressed to J. A. Fodor, Department of Psychology, MIT, E10-034, Cambridge, Mass. 02139, U.S.A.

[1] By 'classical semantic's I'll refer to the tradition of Frege, Tarski, Carnap and contemporary model theory. What this picks out is at best a loose consensus, but it will do for the purposes at hand.

0010-0277/78/0006-0229$2.25

2. Use and mention

Perhaps the *basic* idea of PS is this: providing a compiler for a language is tantamount to specifying a semantic theory for that language. As Johnson-Laird (1977) puts it, the "...artificial languages which are used to communicate programs of instructions to computers, have both a syntax and a semantics. Their syntax consists of rules for writing well-formed programs that a computer can interpret and execute. The semantics consists of the procedures that the computer is instructed to execute (page 189)". Johnson-Laird takes this analogy between compiling and semantic interpretation quite seriously: "...we might speak of the intension of a program as the procedure that is executed when the program is run... (page 192)". And Johnson-Laird is not alone. According to Winograd (1971) "the program [which mediates between an input sentence and its procedural translation] operates on a sentence to produce a representation *of its meaning* in some internal language... (page 409; my emphasis)".

Now, there is no principled reason why this analogy between compiling and semantic interpretation should not be extended from artificial languages (for which compilers actually exist) to natural languages (for which semantic theories are so badly wanted). If compilers really are species of semantic theories, then we could provide a semantics for English if only we could learn to compile *it*. That's the PS strategy in a nutshell.

Since the strategy rests upon the assumption that the compiled (e.g. Machine Language)[2] representation of a sentence is eo ipso a representation of its meaning, it will pay to consider this assumption in detail. It will turn out that there's a sense in which it's true and a sense in which it's not, and that the appearance that PS offers a semantic theory depends largely upon sliding back and forth between the two.

To begin cheerfully: if we take programming languages to be the ones that we are able to compile (thereby excluding natural languages de facto but not in principle), then there *is* a sense in which programming languages are ipso facto semantically interpreted. This is because compiling is translation into

[2] Compiling doesn't, normally, take you *directly* into ML; normally there are inter-levels of representation between the language in which you talk to the machine and the language in which the machine computes. This doesn't matter for our purposes, however, since compiling won't be semantic interpretation unless each of the representations in the series from the input language to ML translates the one immediately previous; and translation is a *transitive* relation. That is: if we can represent English in a high-level language (like PLANNER, say) and if we can represent PLANNER in ML, then we can represent English in ML. Conversely, if our procedural representation of an English sentence eventuates in ML (as it must do if the machine is to operate upon the representation) then we gain nothing in point of semantic interpretation in virtue of having passed through the intermediate representations; though, of course, we may have gained much in point of convenience to the programmer.

ML and ML is itself an interpreted language. If, then, we think of a compiler for PL_i as a semantic theory for PL_i, we are thinking of semantic interpretation as consisting of translation into a semantically interpreted language.

Now, this needn't be as circular as it sounds, since there might be (indeed, there clearly are) two different notions of semantic interpretation in play. When we speak of ML as an interpreted language, we have something like the *classical* notion of interpretation in mind. In this sense of the term, an interpretation specified denotata for the referring expressions, truth conditions for the sentences (or, better, 'compliance conditions', since many of the sentences of ML are imperative) etc. Whereas, when we speak of interpretation for PL_i, we have in mind (not classical interpretation but) translation into a language that is itself classically interpreted (e.g. translation into ML).

It's entirely possible that this is the *right* notion of interpretation for a natural language; that, for one reason or another, it would be a bad research strategy to attempt to classically interpret a natural language 'directly' and a good research strategy to attempt to interpret it 'mediately' (viz. via its translation into some other, classically interpreted, formalism). This idea is quite widespread across the disciplines, the mediating language being called 'logical syntax' in philosophy, 'internal representation' in psychology, and 'ML' in AI. I'm inclined to endorse this strategy, and I'll return to it towards the end of this paper. But what's important for present purposes is this: if we see the PS strategy as an attempt to interpret English by translating it into a classically interpreted ML, then we see straight off that PS isn't an alternative to classical semantics. Rather, PS is parasitic on CS. Translation into ML is semantic interpretation only if ML is itself semantically interpreted, and when we say of ML that it is semantically interpreted, we mean not that ML is translated into a semantically interpreted language, but that ML is *classically* semantically interpreted. Translation has got to stop somewhere.

So, there's *a sense* in which compiling is (or might be) semantic interpretation. Hence, there's *a sense* in which a programming language 'has a semantics' if it is compiled. That's what is right about the PS approach. But there is also a sense in which compilable languages are *not* ipso facto interpreted; a sense in which compiling need not – and normally does not – give the 'intension' of a program. The point is that the interpretation assigned a PL sentence by the compiler is not, normally, its 'intended interpretation'; it's not, normally, the interpretation that specifies *what the sentence means.* One might put it even stronger: machines typically don't know (or care) what the programs that they run are about; all they know (or care about) is how to run their programs. This may sound cryptical or even mystical. It's not. It's merely banal.

You can see the point at issue by considering an example (suggested to me by Prof. Georges Rey, in conversation). Imagine two programs, one of which is a simulation of the Six Day War (so the referring expressions designate, e.g., tank divisions, jet planes, Egyptian soldiers, Moshe Dayan, etc. and the relational terms express bombing, surrounding, commanding, capturing, etc.); and the other of which simulates a chess game (so the referring expressions designate knights, bishops, pawns, etc. and the relational terms express threatening, checking, controlling, taking, etc.). It's a possible (though, of course, unlikely) accident that these programs should be *indistinguishable when compiled;* viz. that the ML counterparts of these programs should be identical, so that the internal career of a machine running one program would be identical, step by step, to that of a machine running the other.

Suppose that these programs were written in English. Then it's possible that the sentence "Israeli infantry surround a tank corps" and the sentence "pawns attack knight" should receive the same ML 'translations'. Yet, of course, the sentences don't mean anything like the same; the compiled versions of the sentences don't, therefore, specify their intensions. Or, if you're dubious about intensions, then: the sentences don't have the same compliance conditions (if they're imperatives) or truth conditions (if they're declaratives); and there is nothing in common between the entities that their referring expressions designate or the relations that their predicates express.[3] In the sense of "interpretation" we are likely to have in mind when we speak pretheoretically of a semantic theory as interpreting English sentences, these sentences are *not* interpreted when they are compiled. Equivalently: the *programmer* knows the interpretation; he knows which interpretation he intends, and, probably, he cares about the program largely because he cares about the (intended) interpretation. But the machine doesn't know the intended interpretation. Nor does it care what the intended interpretation is, *so long as it knows the interpretation in ML.*

How could it be that a compiled sentence is interpreted in one sense yet not interpreted in the other? The answer has partly to do with the nature of computers, and partly to do with the *point* of compiling.

Computers are devices for doing jobs we set; in particular, devices for following orders. What we need for instructing a computer is a language in which we can tell it what jobs we want it to do. What the computer needs to do its job is instructions in a language that it can understand. Machine Language

[3] Well, of course, there has to be *something* in common; the two programs have to be congruent in *some* sense; else why should they go over into the same ML representation? But what is common surely isn't the intended interpretation of the sentences qua sentences of English. It isn't what they *mean.*

satisfies both these conditions. In one way of looking at things, it is a very powerful language since, in a useful sense of "precisely", any job that can be specified precisely can be specified in ML (assuming, as I shall do throughout, that ML is at least rich enough to specify a Turing machine). But, looked at another way – and this is the essential point – it is a very *weak* language. For, the things you can talk about in ML are just the states and processes of the machine (viz. the states and processes that need to be mentioned in specifying its jobs). So, the referring expressions of ML typically name, for example, addresses of the machine, formulate stored or storable at addresses, etc. And the relational terms typically express operations that the machine can perform (e.g. writing down symbols, erasing symbols, comparing symbols, moving symbols from address to address, etc.). Whereas, prima facie, the semantic apparatus of a natural language is incomparably richer. For, by using a natural language, we can refer not just to symbols and operations on them, but also to Moshe Dayan and tank divisions, cabbages, kings, harvestings, and inaugurations. Of course, it's conceptually possible that anything that can be said in a *prima facie* rich language like English can be *translated into* (not just *paired with*) formulae of ML. But, patently, from the mere assertion that English might be compiled in ML and that compiled languages are *ipso facto* interpreted, it does not follow that the meaning of English sentences can be expressed in ML. *A fortiori* it does not follow that a compiler for English would *ipso facto* capture (what I'm calling) the intended semantic interpretations of the sentences of English.

Why, then, do procedural semanticists think that compiling is an interesting candidate for semantic interpretation? I think part of the answer is that, when procedural semanticists talk informally about the compiled representation of English sentences, they allow themselves to forget that ML is interpreted *solely* for machine states, processes, etc. and speak as though we knew how to interpret ML for much richer semantic domains; in fact, for the world at large. That is, they talk as though the semantics of ML constituted a theory of language-and-the world, whereas, in fact, it provides only a theory of language-and-the-insides-of-the-machine.

This equivocation is very widespread indeed in the PS literature. Thus, for example, Winograd remarks that (in PLANNER) "...we can represent such facts as 'Boise is a city'. and 'Noah was the father of Jafeth'. as: (CITY BOISE) (FATHER-OF NOAH JAFETH). Here, BOISE, NOAH and JAFETH are specific objects, CITY is a property which objects can have, and FATHER-OF is a relation" (page 207).[4] To which one wants to reply: 'who says they are?' In particular, neither Winograd (nor PS at large) supply anything remotely

[4] I have left Winograd's citation conventions as I found them.

resembling an account of the relation between a symbol (say, 'BOISE') and a city (say, Boise) such that the former designates the latter in virtue of their standing in that relation. Nor, of course, does the eventual ML translation of 'BOISE IS A CITY' provide the explication that's required. On the contrary, ML has no resources for referring to Boise (or to cities) *at all*. What it *can* do (and *all* that it can do that's relevant to the present purposes) is refer to the *expression* 'BOISE' and say of that expression such things as that it appears at some address in the machine (e.g. at the address labelled by the expression 'CITY'). But, of course, the sentence (capturable in ML) "the expression 'BOISE' appears at the address CITIES" is not remotely a translation of the sentence "Boise is a city". To suppose that it is to commit a notably unsubtle version of the use/mention fallacy.

Or, consider Miller and Johnson-Laird (1976). They provide an extended example (circa page 174) of how a procedural system might cope with the English question: 'Did Lucy bring the dessert?' The basic idea is that the question is "translated" into instructions in accordance with which "[episodic memory] is searched for memories of Lucy..." The memories thus recovered are sorted through for reference to (e.g.) chocolate cakes, and how the device answers the question is determined by which such memories it finds. The details are complicated, but they needn't concern us; our present point is that the authors simply have no right to this loose talk of memories *of* Lucy and internal representations *of* chocolate cakes. This is because nothing is available in their theory to reconstruct such (classical) semantic relations as the one that holds between 'Lucy' and Lucy, or between 'chocolate cake' and chocolate cakes. Strictly speaking, all you can handle in the theory is what you can represent in ML. And all you can represent in ML is an instruction to go to the *address* labelled 'Lucy' (but equally well – and perhaps less misleadingly – labelled, say '#959' or 'BOISE' or 'The Last Rose of Summer') and see if you find at that address a certain formula, viz. the one which is paired, under compiling, with the English predicate 'brought a cake'. Of course, the theorist is welcome – if he's so inclined – to construe the formulae he finds at Lucy (the address) as information about Lucy (the person); e.g. as the information that Lucy brought a cake. But PS provides no account of this aboutness relation, and the machine which realizes the Lucy-program has no access to this construal.

Remember that, though ML is, in the strict sense, an interpreted language,[5] the interpretation of 'Lucy' (the ML expression) yields as denotatum not Lucy-the-girl but only Lucy-the-address (viz. the address that the machine goes to when it is told to go to Lucy and it does what it is told). In short, the semantic relations that we care about, the one between the name and the person and the one between the predicate 'bring the desert' and the property

of being a dessert bringer, just aren't reconstructed *at all* when we represent 'Did Lucy bring dessert?' as a procedure for going-to-Lucy-and-looking-for-formulae. To reconstruct *those* relations, we need a classical semantic theory of names and predicates, which PS doesn't give us and which, to quote the poet, in our case we have not got. In effect, then, a machine can compile 'Did Lucy bring dessert?' and have not the foggiest idea that the sentence asks about whether Lucy brought dessert. For, the ML "translation" of that sentence – the formula which the machine does, as it were, understand – *isn't* about whether Lucy brought dessert. It's about whether a certain formula appears at a certain address.

We get a clearer idea of what has happened if we forget about computers entirely; the brilliance of the technology tends to dazzle the eye. So, suppose somebody said: 'Breakthrough! The semantic interpretation of "Did Napoleon win at Waterloo?" is: *find out whether the sentence "Napoleon won at Waterloo" occurs in the volume with Dewey decimal number XXX,XXX in the 42nd St. branch of the New York City Public Library*'. So far as I can see, the analogy is exact, except that libraries use a rather more primitive storage system than computers do. '"But', giggled Aunt Martha, 'if that was what "Did Napoleon win at Waterloo?" meant, it wouldn't even be a question about *Napoleon*'. 'Aw, shucks', replied Tom Swift".

3. Verificationism

I have stressed the use/mention issue, not because I think that use/mention confusions are what PS is all about, but because, unless one avoids them, one can't get a clear view of what the problems are. However, I doubt that anybody really thinks that you can translate English into ML, taking the latter to be a language just rich enough to specify sequences of machine operations. If you put it *that* way (and that's the way you *should* put it), no procedural semanticist would endorse the program. What is, I believe, widely supposed is that you can translate English into an *enriched* ML; that there is some ML rich enough to say whatever you can say in English, and some machine com-

[5] In what sense is ML 'strictly interpreted'? It's not just that there exists (as it were Platonically) a consistent assignment of its formulae to machine states, processes, operations, etc., but that the machine is *so contructed as to respect that assignment*. So, for example, there is a consistent interpretation of ML under which the formula 'move the tape' is associated with the compliance condition *moving the tape;* and, moreover, it is a fact about the way that the machine is engineered that it does indeed move the tape when it encounters that formula. This parallelism between the causal structure of the machine and the semantics of ML under its intended interpretation is what makes it possible to 'read' the machine's changes of physical state as *computations*. Looked at the other way round, it's what chooses the *intended* interpretations of ML from among the merely *consistent* interpretations of ML.

plicated enough to use that language. So reconstructed, the PS research strategy is to find the language and build the machine.

Of course, you don't want the language to be *too* rich; because (a) it's no news (and no help) that you can translate English into very rich languages like (e.g.) French (or, for that matter, English); and (b) translation isn't interpretation unless it's translation into a (classically) interpreted language, and if ML gets too rich, we won't know how to interpret *it* any more than we now know how to interpret English. That is, to repeat, the very heart of the problem. A compiler would be a semantic theory only if it took us into a very rich ML; i.e., into a language rich enough to paraphrase the sentences of English. But we do not know how to interpret very rich languages, and we do not have anything approaching a clue about what it is to be able to *use* a rich, interpreted language (e.g. what it is to be able to use 'Lucy' to refer to Lucy or what it is to be able to use the internal representation of chairs to think about chairs). PS provides us with no insight at all into either of these questions; indeed, precious few of its practitioners seem to understand that these *are* questions where insight is needed.

Nevertheless, I think that many procedural semanticists think that they have found a middle road. Suppose, in particular, that we equip our machine with sensory transducers and correspondingly enrich ML with names of the transducer input and output states. This enriched ML (which I'll call MLT) still talks about only machine states and processes, but now they are states and processes of something more than a computer: at best a robot and at worst a sort of creepy-feely. Let's assume, also, that MLT has linguistic mechanisms for introducing definitions; e.g. it has a list of dummy names and rules of elimination which allow it to replace complex expressions (including, of course, expressions which contain names of states of the transducers) by single expressions consisting of dummy names. The question is whether the machine language of *this* device is rich enough to translate English.

It seems to me that many, many researchers in AI believe that the answer to this question is 'yes'. Indeed, I think that they had better think that since (a) compiling is translating only when the target language is classically interpreted; and (b) so far, there is no suggestion from PS about how to interpret a target language which is enriched beyond what you get by taking standard ML and adding the names of transducer states (together with the syntactic mechanisms of eliminative definition). So, if the procedural semanticists don't think that English can be paraphrased in MLT, they owe us some alternative account of how the target language is to be classically interpreted. And if they don't think the target language can be classically interpreted, they lose their rationale for taking compilers to be semantic theories in *any* sense of that notion.

Be that as it may, the historical affinities of this program are all too clear. They lie, of course, with the Empiricist assumption that *every* non-logical concept is reducible to sensation concepts (or, in trendier versions, to sensation plus motor concepts) via coordinating definitions. I do not want to comment on this program beside saying that, Heaven knows, the Empiricists tried. Several hundred years of them tried to do without 'thing' language by getting reductions to 'sense datum' language. Several decades of verificationists tried (in very similar spirit) to do without 'theoretical' terms by getting reductions to 'observation' terms. The near-universal consensus was that they failed and that the failure was principled; they failed because it can't be done. (For an illuminating discussion of why it can't be done, see Chisholm (1957)).

My own view, for what it's worth, is that the permissible moral is a good deal stronger: not only that you can't reduce natural language to ML-plus-transducer language, but really, that you can't hardly reduce it at all. I think, that is, that the vocabulary of a language like English is probably not much larger than it needs to be given the expressive power of the language. I think that's probably why, after all those years of work, there are so few good *examples* of definitions of English words (in sensation language or otherwise.) After we say that 'bachelor' means 'unmarried man' (which perhaps it does) and that 'kill' means 'cause to die' (which it certainly does not), where are the *other* discoveries to which reductionist semantics had lead? Surely it should worry us when we find so few cases for our theories to apply to? Surely the world is trying to tell us something? (Of course, there are lots of *bad* examples of definitions; but it's uninteresting that one can get pairings of English words with formulae to which they are *not* synonymous. To get some idea of how hard it is to do the job right, see J. L. Austin's "Three Ways of Spilling Ink", a lucid case study in the non-interdefinability of even semantically closely related terms.)

I want to be fairly clear about what claims are at issue. For the reductionist version of the PS strategy to work, it must be true not only that (a) many words of English are definable, but also that (b) they are definable in a primative basis consisting of ML plus transducer vocabulary; in effect, in sensation language. (a) is dubious but not beyond rational belief. But (b) requires not just that there be definitions, but also that there be an epistemologically interesting *direction* of definition; viz. that the more definitions we apply, the closer we get to MLT. There is, however, no reason *at all* to believe that this is true, and the bankruptcy of the Empiricist program provides considerable prima facie evidence that it is not.

Semantic reduction is the typical PS strategy for interpreting the non-syntactic vocabulary of a rich programming language; viz. semantic reduc-

tion to formulae in MLT. In this respect, there is simply no difference between the typical PS view and that of, say, John Locke. However, as I remarked above, in the standard PS analysis, compiled programs tend to emerge as sequences of *instructions*, and this fact puts a characteristic twist on PS versions of Empiricism. In particular, it burdens PS with a form of verificationism.

For, if 'that's a chair' goes over into instructions, it must be instructions to *do* something. Sometimes procedural semanticists suggest that the instruction is to add (the ML translation of) 'that's a chair' to memory. But it's closer to the spirit of the movement to take it to be an instruction to confirm 'that's a chair', viz. by checking whether that (whatever it is) has the features in terms of which 'chair' is procedurally defined. Indeed, one just about has to take the latter route if one is going to argue that procedures capture intensions, for *nobody* could suppose that 'that's a chair' means 'remember that that's a chair'; where as it's at least *possible* to believe that 'that's a chair' means 'that's something that has the observable features F' where 'F' operationally defines 'chair'. So, once again, it's the verificationism that turns out to be critical to the claim that compiling captures meaning; if 'chair' can't be defined in operational terms, then we have no way of interpreting MLT for chairs. And if we can't interpret MLT for chairs, then we have no way of capturing the meaning of 'that's a chair' in MLT.

Some procedural semanticists admit to being verificationists and some do not. Miller and Johnson-Laird (op. cit.), for example, deny the charge both frequently and vehemently, but the grounds of their denial are obscure. Interpreting liberally, I take it that their point is this: to give procedural reductions of 'chair', 'grandmother', 'dessert', or whatever, you usually need not only transducer vocabulary but also primative terms which express quite abstract notions like 'is possible', 'intends', 'causes' and heaven knows what more. Miller and Johnson-Laird (quite correctly) doubt that such abstract notions can be reconstructed in MLT. Hence they hold that there is no operational analysis for such English sentences as contain words defined in terms of 'intends', 'causes' and the rest. Hence they claim not to be verificationists.

Given a dilemma, you get to choose your horn. Miller and Johnson-Laird do, I think, avoid verificationism, but at an exhorbitant price (as should be evident from the discussion in section 2). In particular, they have no semantic theory for most of the sentences of English. For, consider the sentence E, which goes over into a (compiled) representation of the form'..... causes .....'. The internal expression 'causes' isn't part of ML proper since it isn't interpreted by reference to the states and processes of the machine; and it isn't part of MLT either since, by hypothesis, 'causes' isn't operationally

definable. But the domain in which MLT is semantically interpreted is exhausted by machine states and transducer states. So 'causes' is *un*interpreted in the internal representation of E. So Miller and Johnson-Laird don't have a semantic theory for E.

Some procedural semanticists prefer to be impaled upon the *other* horn. One of the few explicit discussions of the relation between PS and CS is to be found in Woods (1975). Woods takes a classical semantic theory to be a mechanism for projecting the (roughly) truth conditions of syntactically complex sentences from a specification of the semantic properties of their constituent atomic sentences. However "they [classical semantic theories] fall down on the specification of the semantics of the basic 'atomic' propositions (sic; surely Woods means 'atomic sentences') (page 39)." Enter procedural semantics: "In order for an intelligent entity to know the meaning of such sentences it must be the case that it has stored somehow an effective set of criteria for deciding in a given possible world whether such a sentence is true or false (Ibid)." It's a bit unclear how Woods wants his quantifiers ordered here; whether he's claiming that there exists a procedure such that for every possible world....., or just that for every possible world there exists a procedure such that..... Even on the latter reading, however, this must be about the strongest form of verificationism that anybody has ever endorsed anywhere. Consider, for example, such sentences as: 'God exists,' 'positrons are made of quarks,' 'Aristotle liked onions,' 'I will marry an Armenian,' 'the set of English sentences is RE,' 'Procedural semantics is true,' 'there are no secondary qualities,' 'Nixon is a crook,' 'space is four dimensional,' etc. According to Woods, I am, at this very moment and merely in virtue of having learned English, in possession of an algorithm ("an effective set of criteria", yet!) for determining the truth value of each of these sentences. Whereas, or so one would have thought, one can know English and *not* know how to tell whether God exists or what positrons are made of. Good grief, Tom Swift: if all us English speakers know how to tell whether positrons are made of quarks, why doesn't somebody get a grant and find out?[6]

It is of some interest that, having announced that enunciating "such procedures for determining truth or falsity [of atomic sentences]" is the goal of procedural semantics, what Woods actually discusses in the body of his (very interesting) paper is something quite else.

---

[6]Classical verificationism claimed only that, for a sentence to be meaningful, there must *be* (Platonically) a method of verification. This is, of course, much weaker than the present claim that to understand a sentence is to know what the method of verification is.

4. PS as perceptual psychology

Thus far, I've argued as follows:

(a) Compilers won't be semantic theories unless they take us into an interpreted target language.

(b) It's patent that the intended interpretation of English sentences can't be captured in ML proper.

(c) The proposal that we should compile into MLT is the only recommendation that procedural semanticists have made for coping with the semantic poverty of ML.

(d) It's adherence to that suggestion which gives PS its characteristic Empiricist-verificationist cast.

Essentially similar remarks apply to PS treatments of perception; having gone this far, the major theoretical options are, to all intents and purposes, forced. Suppose that F is a formula in MLT such that F expresses the meaning of 'chair'. It presumably follows that determining that 'a is F' is true is a sufficient condition for determining that a is a chair. But now, the (non-syntactic) vocabulary of F is, by hypothesis, exclusively transducer (viz. sensory) vocabulary; so that the normal way of determining that 'a is F' is true would be by reference to the sensory properties of a. However, determining that something is a chair by reference to its sensory properties is a plausible candidate for *perceiving* that it's a chair.[7] So we have an easy bridge from an atomistic view of the semantics of the lexicon ('chair' is proceduraly definable in MLT) to an atomistic view of perception (the 'percept' chair is constructed from sensations; constructed, indeed, by precisely the operations that procedural definitions specify). Semantic reductions thus emerge as perception recipes.

In saying that the options are forced, I don't mean to imply that procedural semanticists generally resist this line of thought; on the contrary, they generally rush to embrace it. The advertising for PS standardly includes the claim that it provides a theory of the interface between language and perception. So far as I can tell, the theory proposed *always* runs along the lines just sketched: semantic decomposition of the lexicon parallels sensory decomposition of percept; the 'translation' of 'chair' into semantic features corresponds to the analysis of chairs into perceptual features.

[7] Of course, not every case of determining that something is a chair by reference to its sensory properties counts as perceiving it; consider the case where somebody tells you what its sensory properties are and you infer from the description that it's a chair. A serious discussion would have to beef up the condition, perhaps by adding that the determination of the sensory properties must involve an appropriate kind of causal excitation of the transducers by a. I'm not, however, attempting to construct an account of perception here; just to indicate how the PS theory arises naturally from the PS treatment of language.

This isn't the place for a diatribe against atomistic views of perception (though I admit to being tempted). Suffice it to make three points.

In the first place, it's notable that – whether or not such views are right – they're remarkably old news. Once again, PS offers no theoretical advance over Locke and Hume (except that, whereas the Empiricists had to use association to stick sensations together, the procedural semanticists can avail themselves of a rather more powerful combinatorial apparatus borrowed from list processing or set theory). I think this point to be worth emphasizing since the flossy technology of computer implementation may make it look as though PS offers an approach to the language-and-perception problem that's importantly novel in kind. To the contrary: if there are reasons for believing the Locke-Hume program was fundamentally wrong about perception, there are the *same* reasons for believing that the PS program is fundamentally wrong about perception; it's the *same* program. Nothing has changed except the gadgets.

The second point is that, so far as I can tell, no new *arguments* for the Locke-Hume program have been forthcoming from the PS camp.[8] This is unsurprising if, as I've been arguing, it's not facts about perception but facts about computers that are forcing the procedural semanticists' hand. Data processes specified in ML won't reconstruct perception unless ML is an interpreted language. And the only interpretation that we currently know how to provide for ML embraces assignments to its terms of (strictly) computational states and relations, together with what can be defined over the names of the transducer states. The perception we can simulate is, therefore, the reduction of complex percepts to transducer outputs. There is, however, no reason for believing that percepts *are* reducible to transducer outputs except that, if they're not, the PS theory of perception won't be true. The fact is that, for some three hundred years, Empiricists have been playing 'heads I win, tails you loose' with the principle that perceptual states reduce to sensory states.[9] The proof that the reductions are possible consists of an appeal to the principle. The proof of the principle is that, were it false, the reductions wouldn't be possible. No workable examples are *ever* given. PS is playing exactly the same game, only it started later.

Now, this last remark may strike you as a smidgen unfair. For – you might ask – doesn't the Winograd program (for example) provide precisely

[8] I say that no *new* arguments have been forthcoming; but there's an *old* argument (in fact, Locke's) which is very much in the air. Viz.: if percepts aren't reducible to sensations, how could concepts be *learned*? A more extensive treatment than this one might have fun tracing the dire consequences of anti-nativism in both classical and PS versions of the Empiricist program (and in much of current cognitive psychology, for that matter.)

[9] See the discussion of Hume's handling of the Empiricist principle in Flew (1964).

the kind of illustrations of successful reductionistic analyses that are here said not to exist?

Answer: no. The whole point about the Winograd program, the trick, as it were, that makes it work, is that the block world that SHRDLU (nominally) lives in is constructed *precisely, so as to satisfy the epistemological and ontological requirements of verificationism;* in particular, each object is identifiable with a set of features, each feature is either elementary or a construct out of the elementary ones, and the elementary features are (by assumption) transducer-detectible. What the Winograd program shows, then is that verificationism is logically possible; there are possible worlds, and possible languages, such that verificationism would be a good semantics for those languages in those worlds (and, *mutatis mutandis*, such that reductionism would be a good theory of the way that percepts are related to sensations in those worlds). The problem, however, is that nobody has ever doubted that verificationism is *consistent*; what there is considerable reason to doubt is that it's *true*. The Winograd program would bear on this latter question if somebody could figure out how to generalize a verificationist semantics for Block World into a verificationist semantics for English. So far, to the best of my knowledge, there are no bids.

The final point is that there *are* alternatives to reductionistic theories of perception, though you'd hardly know it from the PS literature. In particular, it doesn't *follow* from the fact that we are able to recognize (some) chairs (sometimes) that there must be procedures for recognizing chairs, either in the Empiricist sense of sensory check-lists, or for that matter, in any sense *at all*. To see why this is so will require a (very brief) excursis into philosophy of science.

The conventional wisdom in the philosophy of science these days is that there is a principled reason why verificationism won't work. It's the following. The question whether that thing over there is an electron isn't settled by crucial experiment, but by adjudication between the demands that observation and theoretical conservatism jointly impose upon rational belief. We can't give a feature list for 'electron' because, given enough pull from simplicity and conservatism of theory, we may have to decide that it's an electron even if it fails our feature lists, and we may have to decide that it's not one even if it satisfies them. Notice that this is Gestaltism in one sense but not another; there *is* an emphasis on the influence of our *whole* science on the determination of truth value for any given sentence that our science contains. But, *of course*, it doesn't follow that there are no tests relevant to deciding whether something is an electron. On the contrary, this (roughly Quinean) account explains, really for the first time in the philosophy of science, why good experiments are so intellectually satisfying and so hard to

devise. If there were feature lists semantically associated with theoretical terms, then verifying theories would be a relatively dull matter of finding out whether the features are satisfied. Whereas, the present account suggests that *anything we know* (or think we know) could, in principle and given sufficient experimental ingenuity, be brought to bear on the confirmation of any bit of our science. A sufficiently clever experimenter might be able to connect the question whether that's an electron with the annual rainfall in Philadelphia (via, of course, an enormous amount of mediating theory). In which case, more power to him! If the experiment works, we'll have learned something important about electrons (and about Philadelphia).

Now that, of course, is 'philisophy', not 'psychology'. Unless, however, psychology is that way too. It may be that there is no procedural analysis of the concept *chair* precisely because perceptual recognition is fundamentally like scientific problem solving (they are, after all, both means to the fixation of belief.) On this view, in the limiting case, perception would involve bringing to bear our *whole* cognitive resources on the determination of how the world is (what 'perceptual categories' it instantiates). And, of course, every intermediate position is *also* open. It may be that some (but not all) of our cognitive resources are available for perceptual integration; that perceptual integration can be distinguished from problem solving at some interesting point that we don't yet know how to specify. (I think it's likely that this *is* the case; else how explain the persistance of perceptual illusions even when the subject knows that the percept is illusory?) My present point is just that there is a vast range of empirically respectable alternatives to PS atomism. These alternatives suggest the possibility (in some unimaginable and euphoric future) of an integration of the philosophy of science with the psychology of perception, so that 'the logician' can lie down with 'the psychologist' at last. It is, in any event, a reason for worrying about PS that if it *were* philosophy of science, it would be *bad* philosophy of science.

5. What is PS *really* about?

I've thus far argued that PS suffers from: verificationism, operationalism, Empiricism, reductionism, recidivism, atomism, compound fractures of the use/mention distinction, hybris, and a serious misunderstanding of how computers work. What, then is *wrong* with PS? Basically, that it confuses semantic theories with theories of sentence comprehension.

There is a philosophical insight – which goes back at least to Russell and Frege, and, in a certain way of looking at things, perhaps to Aristotle – that can be put like this: the surface form of a sentence is a bad guide to many of

the theoretically interesting aspects of its behavior. If, therefore, you want a theory whose principles formally determine the behavior of a sentence, your best strategy is (a) to pair the sentence with some representation more perspicuous than its surface form; and then (b) to specify the theoretical principles over this more perspicuous representation. So, for example, if you want a theory that formally determines the way a sentence behaves in (valid) arguments, the way to proceed is (a') to pair the sentence with a representation of its 'logical form' (e.g. a paraphrase in a canonical logical notation) and then (b') specify the valid transformations of the sentence over that representation. For reasons too complicated to rehearse here, philosophers have usually taken it that representations of logical form would be appropriate candidates for the domain of rules of (classical) semantic interpretation.

In any event, recent work in the 'cognitive sciences' has added three wrinkles to the basic idea of canonical paraphrase. First, that there's no particular reason why the canonical representation of a sentence should be sensitive *only* to its logico-semantic properties. In principle, we might want a representation of a sentence which captures not just its logical form (in the proprietary sense sketched above) but which also provides an appropriate domain for principles which govern syntactic transformation, or memory storage, or interaction with perceptual information, or learning, or whatever. The more we think of the construction of a theory of canonical paraphrase as a strategy in psychology, the more we shall want to extend and elaborate such constraints. There's no a priori argument that they *can* be satisfied simultaneously, but there's also no a priori argument that they can't. Trade-offs, moreover, are not out of the question; we are open to negotiation.

The second important recent idea about canonical paraphrase is that it might be effected more or less algorithmically: there might be computational principles which associate each sentence with its appropriate canonical representation (or representations, if that's the way things turn out). This is quite different from what Russel and Aristotle had in mind; they were content to leave the mapping from a sentence to its logical form relatively *in*explicit, so long as the logico-semantic behavior of the sentence was formally determined given its canonical representation.

The third idea is that the theory which maps sentences onto their canonical paraphrases (which, as it were, compiles them) might be construed as a model – more or less realistic, and more or less real-time – for what the speaker/hearer does when he *understands* a sentence. That is, we might think of a speaker/hearer as, to all intents and purposes, a function from canonical paraphrases (taken, now, as mental representations) onto forms of utterance. Patently, the more kinds of constraints internal representations can be made

to satisfy qua domains for mental operations, and the more real-time-like the function from canonical paraphrases to surface forms can be shown to be, the more reason we shall have to take this sort of model of speaker/hearers as empirically plausible.

I take it that this general approach is common to *all* current work on cognition, barring only Gibsonians and eccentrics.[10] That is, it's common ground to AI, PS, linguistics, cognitive psychology, psycholinguistics, and for that matter, me. What's special about PS is only a bundle of proclivities within this strategic commitment: procedural semanticists tend to emphasize that canonical paraphrases should be constrained to provide domains for principles which specify the interactions between sentential and contextual information; they tend to emphasize real-time constraints on the recovery and coding of canonical representations; they tend to countenance trade-offs which buy feasibility at the price of generality; they tend to be relatively uninterested in constraining canonical representations by considerations of linguistic plausibility. Such proclivities amount, in effect, to an implicit research strategy within the general cognitivist camp. Like any such strategy, it is to be judged by its pay-off. My own view, for what it's worth, is that the pay-off has, thus far, been negligible; that, so far, practically nothing has been learned about language processes from the PS models, whereas quite a lot has been learned by investigators who have taken opposed views of how a theory of internal representation might best be constrained. That, however, is as it may be.

My present point is two-fold. First, if I'm right about what PS and the rest of us are up to, then there's no new account of language (or of language-and-the-mind) explicit or implicit in PS; there's simply a set of recommended tactics for approaching the problem we *all* take ourselves to be dealing with: how should we best model the speaker/hearer, given the assumption that the speaker/hearer is to be viewed as a function from utterances onto internal representations. That is what we *all* are doing is trying to provide *a model of sentence comprehension* (/production); a model which says (a) what the speaker/hearer has in his head insofar as having *that* in his head constitutes understanding a sentence; and (b) which explains how whatever he has in his head when he understands a sentence manages to get there.

The second, and final, point is that what *none* of us are doing (including NB, PS) is providing a semantics for a natural (or any other) language: a theory of language-and-the-world. What we're all doing is really a kind of logical syntax (only psychologized); and we all very much hope that when we've got a reasonable internal language (a formalism for writing down

[10]For a development of this theme, see Fodor (1975)

canonical representations) someone very nice and very clever will turn up and show us how to interpret it; how to provide it with a semantics. What has disguised this fact from the procedural semanticists is that everybody else has given up supposing that the semantics will be verificationist. This difference *makes* a difference, as we've seen. In particular, it has allowed the procedural semanticists to suppose that a theory of how you understand a sentence can do double duty as a theory of what the sentence means; to confuse compiling with semantic interpretation, in short.

Whereas, because the rest of us are *not* verificationists, we can live with the fact that 'chair' refers to chairs; we don't have to go around supposing that 'chair' refers to bundles of sense data. It is, of course, not very *interesting* to say that 'chair' refers to chairs, since we have no *theory* of reference and we have no *mechanism* to realize the theory.[11] A fortiori, we don't know how to build a robot which can *use* 'chair' to refer to chairs. But, though "chair' refers to chairs' isn't interesting, we don't mind that so much since reference isn't what we're working on. We're working on logical syntax (psychologized). Moreover, "chair' refers to chairs' has one striking advantage over 'chairs are made of sense data'; it's not interesting, but at least it's *true*.

So, Tom Swift, here is how things stand: understanding a natural language sentence *may* be sort of like compiling. That is, it may be a matter of translating from the natural language into a paraphrase in some canonical target language. If so, the target language must have a very rich semantics (nothing like MLT) and must be syntactically perspicuous in a way that natural languages are not. Nobody has the foggiest idea of how to connect this sys-

---

[11] I'am distinguishing between (a) a 'theory of reference' (and, more generally, a classical semantic theory) which consists of a function from the expressions of a language onto the objects which interpret them; and (b) a theory of the mechanism which realizes the semantics, viz. the kind of psychological theory which answers the question: 'what about a given language, or about the way that a given organism uses it, makes one or another semantic interpretation the right one for that language?' (a) and (b) *both* differ from (c) theories that operate in the area that I've called psychologized logical syntax.

Theories of type (a) are familiar from work in classical semantics. Theories of type (c) are what people generally have in mind when they talk about "internal" ("canonical", "mental") representation (hence all varieties of PS, properly construed, belong to type (c), as does practically all of modern linguistics and cognitive psychology). There are, to the best of my knowledge, no *plausible* theories of type (b), though there are plenty of *im*plausible ones. For example, Skinnerian theory is best considered an attempt to answer the question: 'what about the relation between an organism, an utterance form, and an object makes the use of the second by the first constitute a reference to the third?", the proposal being that the utterance form refers to the object when and only when the object is a discriminative stimulus for which the utterance is a discriminated response.

Perhaps it goes without saying that the desirable situation is the one where the formal semantics (type a), the account of the logical syntax of the vehicles of representation (type c) and the psychology of reference (type b) all fit together.

tem to the world (how to do the semantics of internal representations) but that's OK because there are lots of other constraints that we can impose and maybe even meet; constraints inherited from logic, linguistics, psychology and the theory of real-time computation. It might be instructive to try to build a machine which meets some of these constraints. Qua computer, such a machine would carry out sequences of operations upon the (de facto uninterpreted) formulae of some canonical language. Qua simulation, it would provide a psychological model insofar as mental processes can be construed as sequences of formal operations upon mental representations. But do not, Tom Swift, mistake this machine for a semantic theory. *A fortiori, DO NOT MISTAKE IT FOR YOUR GRANDMOTHER.*

Right, Tom Swift: back to the drawing board.[12]

References

Austin, John L. (1966) "Three Ways of Spilling Ink." *Philos. Review, 75* (4).

Bobrow, Daniel G. and Collins, Alan (Eds.) (1975) *Representation and Understanding,* Academic Press, New York.

Chisholm, R. (1957) *Perceiving,* Cornell University Press, Ithaca.

Flew, A. (1964) "Hume" in O'Connor (1964).

Fodor, Jerry A. (1975) *The Language of Thought,* Thomas Y. Crowell Co., New York.

Johnson-Laird, P. (1977) "Procedural Semantics, *Cog., 5,* (3), pp. 189–214.

Miller, G. and Johnson-Laird, P. (1976) *Language and Perception,* Harvard University Press, Cambridge.

O'Connor, D. J. (ed.) (1964) *A Critical History of Western Philosophy,* The Free Press of Glencoe, London.

Winograd, T. (1971) *Procedures as a Representation for Data in A Computer Program for Understanding Natural Language,* M.I.T. Project Mac, Cambridge.

Woods, W. (1975) "What's in A Link," in Bobrow and Collins (1975).

[12]I wish to thank the many members of the AI community who read the manuscript and offered good advice (like "My God, you can't publish *that*!"). I'm especially indebted to Steven Isard, Philip Johnson-Laird and Zenon Pylyshyn for illuminating comments on an earlier draft.