"Should computer scientists read Derrida?"

Wesley Phoa
School of Computer Science and Engineering
University of NSW

DRAFT of 24/5/93

Abstract

A quarter of a century ago, Derrida made the remarkable claim that his work should be the foundation for AI and cognitive science. This paper attempts to follow up that lead. It starts by giving two examples of philosophical positions which have had a practical impact on computing—positions which Derrida has criticised. Next comes an outline of some aspects of Derrida's work. The final section discusses some speculative examples of how Derrida's ideas might actually be relevant to computing. It closes by offering an amateur's deconstructive reading of a paper by Peter Slezak.

Introduction

Should computer scientists read Derrida? The answer is obviously 'no'; but I want to argue that it's not *immediately* obvious that the answer is no. After all, Derrida thinks we should. He writes, above all, about *writing*—language, representation—and

> ...whether it has essential limits or not, the entire field covered by the cybernetic *program* will be the field of writing. If the theory of cybernetics is by itself to oust all metaphysical concepts—including the concepts of soul, of life, of value, of choice, of memory—which until recently served to separate the machine from man, it must conserve the notion of writing...[1]

And it is arguable that computer scientists should at least read *about* Derrida. Since Derrida is a controversial figure (to put it mildly) even in philosophical and literary circles, this may seem an implausible statement. But I think it can be defended; and I will do my best, as a relative newcomer to both Derrida and computer science, to do so.

My claim is not that Derrida has come up with specific technical innovations which are directly applicable in AI. Nor do I claim that his ideas form a comprehensive basis for a new approach to cognitive science, though others might want to claim this. All I propose is that reading Derrida, or reading about Derrida, might encourage us to look at various general issues in computer science in a new way, and put us in the right frame of mind to appreciate certain problems, to understand the shortcomings of certain approaches, to see what is going on beneath the surface of certain debates, and to get around certain obstacles.

[1] [Derrida 1967], p. 9. The idea that the fundamental issues of computer science tend to depend on questions about the nature of language is by now familiar—see [Winograd and Flores 1986]—though by no means unquestioned.

Why to study philosophy Computer science departments are usually divided into three groups: a tiny band of philosophically literate people (often socially isolated and displaying distinctive and unusual physical characteristics); a slightly larger group of people who think philosophy is kind of interesting and important but know less about it than they'd like to think (like me); and the majority, who believe that computer scientists should get on with their lives, i.e. write code. As a gesture to this third group I begin with some general remarks.

Why should we study philosophy? There is an obvious reason:

> Practical men, who believe themselves to be quite exempt from any intellectual influences, are usually the slaves of some defunct economist.[2]

Replace 'intellectual' with 'theoretical' and 'economist' with 'philosopher', and we have a good description of the situation in computer science. All practical work is based on philosophical presuppositions: they may be conscious or unconscious, innocuous or fatal. In AI, some people might argue that they are often fatal.[3] In any case, we might as well be aware of them, and aware of the alternatives.

There is a more positive reason to be interested in philosophy, though: philosophical reflection can be a potential source of practical ideas. For example, there is an alternative to the usual notion of 'meaning': that the meaning of a term is given by a fixed definition. We can instead think of a term as having meaning insofar as it differs in various ways from other terms in the same discourse; meaning is now something that evolves as differences are elaborated and the discourse broadens and deepens—for example, the meaning of 'sadness' changes once we distinguish it from 'depression'. This idea has been useful in building real systems that perform well, cf. [Gaines and Shaw 1990]. And though it is usually attributed to Kelly, a psychologist, it can really be traced much further back, to the philosopher/linguist Ferdinand de Saussure. (Who incidentally stands at the beginning of a road running into and past the territory of structuralism; Derrida himself, though offering a powerful critique of Saussure, acknowledges an intellectual debt.)[4]

Of course, there have always been plenty of philosophically literate people in the AI and cognitive science communities. But up till quite recently, Anglo-American analytic philosophy has more or less had a monopoly on the terms of the debate. Whether or not other traditions have a role to play seems to be a rather controversial question; the appearance of continental philosophy in [Winograd and Flores 1986] (more specifically, the ideas of Heidegger and Gadamer) seems to have sparked off some vigorous arguments.[5] Poststructuralism has not had much impact so far, though [Gaines 1991] does exhort cognitive scientists to study Foucault and Derrida,[6] and [Wilson 1992] discusses, rather sketchily, the relationship between Derrida's reading of Freud and connectionist theories of cognition in psychology.

---

[2] [Keynes 1936], p. 383.

[3] This is certainly a claim made in [Winograd and Flores 1986].

[4] There are many such resonances, mostly less direct. For example the analysis of O. Henry's "Gift of the Magi" in [Winston 1981] vs. the analysis of the first sequence of *Goldfinger* in "Introduction to the structural analysis of narratives" [Barthes 1966].

[5] One might expect recent attempts to import Buddhist philosophy into cognitive science to be met with even more skepticism. Clearly the time is not yet ripe for a paper called, "Should computer scientists read Hakuin?", or, God forbid, "Should computer scientists read Fritjof Capra?"

[6] My impression is that, although Foucault can offer us general insights and a useful (re)orientation towards our discipline, Derrida's work bears more directly on the problems of computer science.

How to study Derrida Condemning a continental philosopher for being unintelligible is like condemning a fishmonger for being smelly; it's part of the job. Luckily there is a whole industry of Anglo-Americans explaining continental philosophy to other Anglo-Americans. The explanations may not be very good, and they are often incomplete or biased, but they are still useful. And they are generally phrased in language that is acceptable to readers with an analytic bias—something which most of us English-speakers have, consciously or not.

The most useful source, I think, are the three essays on Derrida in [Rorty 1983–90] (and presumably Rorty's other essays on Derrida, which I haven't had the chance to read yet); these do assume some basic familiarity with the primary texts. Although Rorty's attempt to assimilate deconstruction into the American pragmatist tradition involves a little distortion, and is not ultimately wholly satisfactory, it is an excellent way to get into Derrida. I have also found the more introductory account in [Wood 1990] quite helpful, though it is less down-to-earth.

Derrida, of course, is better known and appreciated in English departments than in philosophy departments. I am not familiar with much of the literature generated by the practitioners of (the cruder form of American-style) deconstruction. My impression is that they produce, at best, thought-provoking coffee-table books.[7] At worst, their writings are extremely pretentious and, to the extent that they are not incomprehensible, disappointingly shallow.[8] There seem to be a few good expositors among Derrida's more thoughtful followers, though: Norris, for example, gives a helpful account of Derrida in [Norris 1987] and [Norris 1982], though he still doesn't present *arguments* in a way that would be satisfactory to the average computer scientist, for example.

I should recommend some works by Derrida himself. This is a much easier task: apart from some random passages here and there, the only ones I've read are [Derrida 1967] and [Derrida 1972/77/88]. The former is a key text, but very heavy going; the latter is unexpectedly lucid, and extremely entertaining—the afterword, in particular, is very enlightening. But anything by Derrida is bound to be frustrating reading for a computer scientist who isn't just reading for fun.

Acknowledgements The idea of this talk grew out of some enlightening discussions with Amitavo Islam (my philosophical consultant) and Paul Compton; in particular, Amitavo Islam raised and clarified innumerable issues, and Paul Compton provided me with useful references and some perspective and orientation. Those two people cannot, however, be held in any way responsible for the ideas expressed here.

I have also had helpful meetings with Clark Quinn, who kindly lent me his copy of [Winograd and Flores 1986]; I had not realised how many points of contact there would be (and I make very little attempt to explore them here). I was financially supported by an ARC Postdoctoral Research Fellowship.

Two case studies

Before discussing Derrida, I will give two examples of how philosophical ideas can have an impact on computer science in practical ways. I chose these two particular examples

[7] Avital Ronell is a good example. Derrida does this too, however: cf. *Glas*, *Cinders*.

[8] Though Rorty points out that they have still been extremely effective politically, often in very positive ways: see "De Man and the American Cultural Left", in [Rorty 1983–90].

because they illustrate the further point that philosophy can influence practice at some distance, so that this influence is barely perceived or acknowledged; and also because Derrida has offered a critique of both of these positions.

First an important disclaimer. To say that a body of practical work can be traced back to certain philosophical ideas is not to say that its validity depends on the validity of those ideas. For instance, I will claim that some recent work on theorem provers can be traced back to Husserl. I am not claiming that, by discrediting Husserl, one would show that this work was valueless; it could presumably be justified, philosophically, in other ways—if it needs such a justification. Rather, I am only claiming that some of Husserl's ideas and attitudes, and his general mindset, have been inherited (unconsciously, and in a mutated form) by the people who design and use these theorem provers, and that this fact shapes their work to a great extent.

The early Wittgenstein

I begin with the early Wittgenstein. Actually, what I am about to summarise is not the position expounded in [Wittgenstein 1921], but a sketchier collection of views which are associated with the Wittgenstein of that time, some of which he inherited, some of which he arrived at himself. I think it is this looser 'position'—the general features of his *a*psychologistic theory of meaning—that has had enduring influence, rather than the specific metaphysics of language put forward in the *Tractatus*. In a sense, I only attach Wittgenstein's name to it as a matter of convenience.

The early Wittgenstein sees the world as consisting of elementary propositions or states of affairs: facts. The role of language is to describe these states of affairs. A proposition does this in the following way: its linguistic elements correspond to particular things[9] in the world, and its logical form—the logical relationship between those elements—mirrors the relationship between those things, i.e. the logical form of the state of affairs being described. This is roughly what is meant by the slogan, "A proposition is a picture of reality".

The sole connection between language and the world is the way in which its elementary terms denote or signify things in the world. The way in which these terms are combined to form propositions—syntax—and the way in which one infers propositions from other propositions—logic—are 'purely logical', entirely independent of the world, or (to put it another way) independent of semantic considerations.

Wittgenstein is describing an ideal language in which terms have an unambiguous denotation and in which logical form is transparent. The mechanism by which terms can denote things remains unanalysed. Syntax is specified by a collection of formal rules that do not refer to the meanings of terms. Propositions are true or false. Propositions which contain quantifiers are analysed into (very long) ones which do not. A logical deduction is regarded as sound exactly when it is a tautology; and Wittgenstein invented truth-tables in order to give a criterion for deciding when something is a tautology.[10]

It is easy to see how influential these views have been: for example, they form the

[9] I am glossing over a subtle point here. It seems that the 'things' of the *Tractatus* are not 'positive concrete entities' apprehended by the senses as Russell might have understood them, but defined in terms of their occurrence in possible states of affairs: see D. O'Brien, "The unity of Wittgenstein's thought", in [Fann 1967]. However this point does not seem to be relevant to our present discussion.

[10] He regarded Frege's axiomatic method as arbitrary and unsatisfactory.

basis of Newell and Simon's physical symbol hypothesis.[11] Another example: Chomsky's 'Autonomous Syntax' thesis, that a firm line can and should be drawn between syntactic and semantic considerations, forms the basis for most formal work on the syntax of natural language, and certainly for the picture of linguistics presented to undergraduates: see [Radford 1981]. Admittedly, Chomsky only puts this forward as an 'empirical hypothesis', and indeed it is arguably not even a good one.[12] But it seems to transform itself easily into a tacit and unquestioned assumption.

Another example is our approach[13] to knowledge representation. For instance, a simple knowledge base in Prolog might be set up as follows: we identify the 'real-world' objects and relations we wish to represent; code them up in Prolog syntax (so there is an unambiguous correspondence between symbols and 'real entities); and then allow the interpreter to make deductions about them in a purely formal, mechanical way. And our usual understanding of logic programming is in terms of a classical semantics.[14]

Wittgenstein's early views also seem to be echoed in the logicist position in cognitive science. The idea that cognition consists of manipulating mental representations (similar in character to the ones we use on a conscious level) via precise logical rules (similar to those of formal logic) is characteristic of Wittgenstein in this phase—though it was certainly not original to him. At the end of the essay, we will return to this notion of cognition.

Finally, the view that the validity of logical and mathematical arguments reduce to considerations about truth and falsity and pure logical form, and do not rely on an account of mental processes, is reflected in certain current approaches to automated theorem proving. The emphasis on truth-functional semantics and on decision procedures for logics, and on 'black box' theorem provers which can be given a problem and left to chug away independently—eventually coming up with the answer 'yes', or 'no'—arguably stems from this attitude about logic.

The phenomenological programme

A rather crude summary of Husserl's programme would be: our own experience, and our own perceptions, are the only reliable basis of knowledge. In dealing with foundational matters we should speak only in terms of our own consciousness, and 'bracket out' everything else (such as any talk of an 'outside world' existing independently of perception). But to evade the charge that this leads to a purely subjective or psychologistic theory, he proposes that we study our *common* experience of consciousness: the analysis should be of an *ideal* consciousness experienced by a transcendental subject.

Husserl's phenomenology is reflected in Brouwer's approach to the foundations of mathematics. This intuitionistic line says that mathematics is not about abstract logical laws, or meaningless formal games, or Platonic mathematical entities existing independently

[11] See [Compton and Jansen 1990], p. 242.

[12] For example, Radford points out that there are good semantic arguments (based on 'selection restrictions' shared by pairs of sentences) for positing the existence of, say, a WH-MOVEMENT transformation rule in English (which would explain these shared restrictions by relating these sentences): p. 165–167. It would be said to have to rule such arguments out as inadmissible.

[13] Presented to undergraduates without much critical reflection in books such as [Brathe 1990].

[14] An alternative view, based on proof theory, has recently been gaining popularity. This is an example of the possibility, alluded to above, that the philosophical foundations of a body of practical work can change; however, it also demonstrates that such a change will probably have pragmatic consequences.

of human subjects; mathematical assertions are assertions about mental constructions. For example, there is no external, complete 'set' of natural numbers; our notion of the sequence of natural numbers is derived from our experience of successive states in time, and our intuition that these states can proceed indefinitely.

Such views have well-known consequences for mathematics. However, I am more concerned here with Brouwer's influence in computer science. This influence stems largely from the various attempts to formalise intuitionistic mathematics—an ironical fact, in the light of Brouwer's hostility to formal systems.

Let me recall the two kinds of theorem-provers that exist for higher-order logic. I don't know of any useful fully automated theorem provers for higher-order logic; this must be partly because it doesn't seem to have any interesting decidable fragments. The systems that exist are *proof assistants*, which let a user construct a proof interactively with some assistance from the machine. And loosely speaking, these can be divided into *tactics-based* theorem provers (like HOL and Isabelle) and theorem provers which allow users to construct explicit *proof-terms* (as in NuPRL and Lego).

The latter are mainly based on the formal systems of Per Martin-Löf, which were intended to formalise intuitionistic mathematics. Unlike the better-known systems familiar from the work of, e.g., Heyting and Gentzen—which simply let us determine which propositions are derivable—Martin-Löf's systems contain an explicit notation for proofs. That is, one writes down typed λ-terms which represent the mental constructions one performs in carrying out a proof. To express the point of view in a slogan: Martin-Löf type theory is the language of the transcendental subject.[15]

It would be oversimplifying matters to call Martin-Löf a phenomenologist, but the line of descent is clear. And the people who use these theorem provers seem genuinely to regard them as tools for representing mental constructions, or at least those abstract features of mental constructions which have some universal significance. This fact explains many of the attitudes prevalent in that community: for example, the suspicion of other kinds of theorem provers seems to be based on their failure to represent this kind of mental information—a proof that relies on the correctness of a piece of code (a decision procedure, or a tactic) is seen as less reliable than a proof which consists of a formal record of a mental construction which one has performed oneself, albeit with machine assistance.[16]

I should say that this may have practical significance beyond the world of theorem proving. Martin-Löf type theory has been seriously proposed as a suitable framework for formal specification and software development—bringing with them some entirely different attitudes about the nature of specification and refinement[17]—and even Brouwer's choice sequences may find their way into computing, via Martin-Löf's recent work on streams.

Interestingly, at least one tactics-based higher-order logic theorem prover—Abstract Hardware Limited's LAMBDA system—can be traced back to Brouwer's intuitionism via topos theory, the categorical formulation of intuitionistic higher-order logic. However, it must be admitted that in this case the philosophy was lost somewhere along the way—probably at the marketing stage—and in fact the current version of LAMBDA is classical, not intuitionistic.

---

[15] Of course, there have been other attempts at formalising the notion of 'construction', such as the earlier theories of Goodman.

[16] This is not to say that non-philosophical factors such as professional jealousy are irrelevant; though in this case, my personal impression is that they are secondary.

[17] See for example [Thompson 1991]

What happened to them?

So these two philosophical positions are, decades later, having a practical influence on the way some of us do computing. It might therefore be worth remarking that neither is really respectable any more. Wittgenstein changed his mind, of course, and as we shall see his later position resembles Derrida's in several important ways; while Husserl's phenomenology has been subjected to a number of different critiques, Derrida's among them.[18]

In each case the 'hole' has to do with the nature of representation. The process by which symbols denote things was effectively taken as unproblematic by the early Wittgenstein, as it was by his intellectual predecessors such as Frege and Russell; while in the case of phenomenology, the whole notion of a transcendental subject relies on on our being able to compare our experiences, i.e. represent our mental lives faithfully using language. And this is where Derrida comes in.

A potted Derrida

> As writing, communication (if we bother to retain this word) is not the means of transportation of meaning, the exchange of intentions and meanings, discourse or the "communication of minds"... the system of speech, of consciousness, of meaning, of presence, and of truth, etc., will only be an effect and should be analyzed as such.[19]

As most of you will be even more unfamiliar than I am with Derrida's texts, I should feel obliged to give some kind of introduction to them. It is standard to make a disclaimer at this point, perhaps saying that 'Derrida's' 'ideas', to the extent that 'he' 'has' any, can't be summarised; but also that such a summary is anyway permissible, being nothing more than another example of the free (differantial) play of signifiers, licenced by Derrida's texts themselves. I'll take all that as read, then, and proceed.

We can think of Derrida as saying something about language, or communication, or representation. But each of these words has a certain misleading connotation. It might be better to say that Derrida is concerned with the *nature of discourse*, and that his analysis has certain broader consequences (such as undermining the basis of philosophy).

There are things about Derrida which are difficult to come to terms with at first. One of the main ones is his style, which is definitely continental. In one sense this is superficial—if he had been brought up in Britain he would probably write like the later Wittgenstein, or rather, like a cheerful Wittgenstein, if such a thing can be imagined—though we shall see that one of Derrida's lessons is that we cannot separate form from content.

The comparison with Wittgenstein is appropriate (and, by the way, not at all original) for another reason: both of them *demonstrate*, by means of certain expository strategies, certain things which they feel unwilling or unable to *state*. This is what lies behind the peculiarly rambling impression left by [Wittgenstein 1953], and the disorienting mixture of logical argument and wordplay/wilful misinterpretation/playful improvisation that makes up nearly all of Derrida's texts. I will return to this point later.

---

[18] Indeed, Derrida's first published work was a translation of Husserl's *Essay on the Origin of Geometry*, with an introduction considerably longer than the work itself.

[19] J. Derrida, *Marges de Philosophe*, p. 392; cited in [Fekete 1984], p. 223.

Speech and writing

Derrida's starting-point is the contrast between speech and writing. Speech is characterised by *presence:* one person says something to another, with a definite intention and within a fixed context, and the utterance thus communicates a determinate meaning. Writing, on the other hand, is characterised by *absence:* it consists merely of symbols, cut off from the person who wrote them and the context in which they were written, and without any meaning in themselves; the meaning of a text is indeterminate, depending entirely on the surrounding context.

Derrida contends that in the traditional, 'logocentric' way of thinking, speech is privileged over writing. In other words, the notion of determinate ideas being transmitted via spoken language has always been taken to be the ideal of discourse; writing has always been regarded as an imperfect and parasitic supplement to speech. But, he replies, it is actually the notion of writing, with its symbols of shifting and uncertain meaning, that is central.

The point, however, is not that "speech is generally regarded as better than writing but actually it's the other way around". It is a little more subtle: namely, that our conception of speech is commonly accepted as the paradigm of discourse, but that the notion of writing should take its place. In other words, 'writing', for Derrida, has a vastly extended range of meanings: it encompasses spoken and written language, music, photography, film, culture, history, genetic code... Calling all these things 'writing' is meant to draw attention to the fact that in all these cases, symbols cannot be regarded as having a fixed meaning which we can get at if we could only decode them; instead, the 'meaning' of a symbol relies wholly on its context, on the network of *differences* between it and the other symbols in play in the context, and hence its meaning is also endlessly *deferred*, dependent on the further interpretation of those other symbols.

> ...[deconstruction is] the effort to take this limitless context into account, to pay the sharpest and broadest attention possible to context, and thus to an incessant movement of recontextualisation. The phrase... "there is nothing outside the text", means nothing else: there is nothing outside context.[20]

Derrida does not claim that his own writings transcend this 'incessant movement'. The idea that one can step back and talk about language from the outside, in some definitive way, is a delusion; one is always trapped by the fact that one's own language cannot say anything timeless and absolute, because its meaning depends completely on the context in which it is read. Philosophical writing cannot do this, because it is still writing, after all, and still subject to the same indeterminacy and play of meaning (as Derrida tries to demonstrate again and again). We will see later that this commits Derrida to using some unusual strategies to 'make' his 'point'.

Deconstruction, however, is not simply relativism. To say that meaning is essentially indeterminate does not mean there is no such thing as meaning. On the contrary,

> ...to the extent to which it... is itself rooted in a given context,... [deconstruction] does not renounce... the "values" that are dominant in this context (for example, that of truth, etc.).[21]

---

[20] "Toward an Ethic of Discussion", [Derrida 1972/77/88], p. 136.

[21] "Toward an Ethic of Discussion", [Derrida 1972/77/88], p. 137. We can see why deconstruction can sometimes seem extremely radical, while at other times appearing rather conservative. In fact one of the attractions of being a follower of Derrida is that one can be a conservative and still feel like a radical.

In other words our common use of words and ways of reasoning only break down when we start to push at the limits of a context, to try and say something permanent, universal or transcendental. It is quite clear that in ordinary life we normally succeed in saying what we mean and people normally understand us. Slippages of meaning, though they always occur, are most serious when we try to push language beyond its day-to-day usage: for example, when we try to use it to formulate a *general* theory of something-or-other.

In other words, Derrida isn't saying that you can't say anything; he's just saying that you can't ever say an absolutely specific something, that you don't have total control over the interpretation of your own utterances or writings; that what seems clear and unambiguous can always become enigmatic. This in itself is not disastrous. It demolishes some philosophers' programmes; but so, in a different way, did Gödel's theorem, and though that was depressing/exhilarating at the time, now it just doesn't seem that shocking.

Applicability to formal languages I suspend this discussion briefly to dispose of a red herring. One might argue that the foregoing remarks apply perfectly well to natural languages, which are by nature imprecise, but do not apply to formal languages whose symbols each have a completely precise meaning. This is missing Derrida's point. It is true that we can manipulate expressions in a formal language in a precise and determinate way *so long as they remain completely uninterpreted* (though the more radical deconstructionists might have something to say even here—about how 'rules' will be construed in different contexts, etc.). However, once we try to *use* a formal language, to give some meaning to its symbols via our own language, exactly the same problems arise.

Implications for a theory of meaning

There is no room in these notes to try and justify Derrida's attitude. Many of his texts seem to have this end: they are critiques of some philosophical position—phenomenology, speech-act theory, Saussure's structural linguistics, Rousseau's writings on the origin of language—in which Derrida shows how superficially plausible ideas turn out to rely on logocentric notions of discourse, discourse as speech rather than writing, and this reliance turns out to have paradoxical results.

In the rest of this section we will, instead, sketch some consequences of adopting this attitude towards language. One of the interesting ones is its implications for what a theory of meaning can be. This is the subject of the essay "Signature Event Context", which triggered off the exchange chronicled in [Derrida 1972/77/88]. In this essay, Derrida offers a critique of Austin's speech-act theory—an exercise which, in the light of [Winograd and Flores 1986], might be of particular interest to computer scientists.[22] Derrida is not criticising Austin's arguments as such; rather, he is saying that, though they are quite sound, they cannot be taken as the basis of a theory of meaning.

Austin essentially suggests that the meaning of an utterance can be determined by looking at what kind of action it is (an assertion, a command, a promise...); this will involve looking at the intention of the speaker, as indicated by the speaker's actions within the surrounding context. In following this idea up, he excludes (for the moment) 'non-serious' discourse: where the speaker is joking, or acting in a play, and so on. This is

[22] Winograd and Flores also quote Habermas quite approvingly; this is another interesting point of contact with the present essay, since Habermas has been a trenchant critic of Derrida. I have not begun to work these issues out.

perfectly reasonable. Derrida's point, however, is that although this exclusion is reasonable, it is exactly what makes it impossible for Austin to give a general account of meaning rather than simply a useful account of what certain utterances mean in certain situations.

The argument seems to be roughly as follows. The essential feature of language is that utterances (or strings of written symbols) are *iterable:* they can be repeated in different contexts, both serious and non-serious. There is no utterance that can only be used in a serious context; such a thing would not constitute language as we understand it. Therefore it is an essential feature of language that it can be used in all kinds of contexts, serious and non-serious. A theory of meaning must take into account this essential structural feature of language. It is therefore misguided to try to define meaning by first restricting ourselves to 'serious' contexts.

What is the alternative, then? It is *not* to deny the importance of the notion of 'intention', or of 'serious context'. All Derrida denies is that these notions should be at the centre of a theory. Rather, an account of meaning should be an account of the function of an utterance or a text in different contexts; in such an account, aspects of intentionality or seriousness (within a given context) will still have an important role to play—but not a determining role:

> ... intention is not annulled... but rather *inscribed* within a system which it no longer dominates.[23]

The end of the book

To repeat, Derrida seems to claim that

- Meaning is *entirely* dependent on context; and
- Meaning can never be *fully* specified by a given context.

The first claim is reminiscent of the later Wittgenstein: the meaning of a word depends entirely on the language-game within which it is being used. But the second claim goes further: we can never specify exactly which game we are playing, exactly what the rules are—any attempt to do so simply pushes the question back one more level.

This has important consequences. It means that the notion of a book as a self-contained repository of meaning is an impossible one. Again the word 'book' is used in a very general way, to refer not just to a physical text, but to any fixed (textual, cultural or historical) context. Meaning can never be contained no matter how wide we draw the boundaries. Any attempt to close it off, though it may seem to succeed while we are operating comfortably within the bounds of a given context, will always break down if we push the logic of the situation far enough. The boundary of a book takes the form of a Klein bottle.

It is important to recognise the positive outcome of this train of thought. In discarding the notion of 'book', of a comprehended totality—a self-sufficient island of meaning—we open up discourse to a rich and unending play of associations and displacements. We will see an example of this later.

---

[23] [Derrida 1967], p. 248.

Deconstructing binary oppositions

I have sketched what I think are some of Derrida's most important ideas. Just as important, however, are his strategies: those he uses in the process of reading and interpreting texts, and those he deploys in structuring his own texts. Like the later Wittgenstein, Derrida denies that he is laying down a general theory or a body of doctrine; unlike Wittgenstein, he even denies that he is setting out a general method; instead, he has positions, insights, attitudes—strategies.

One of the most important of these strategies lies behind the common view of deconstruction, that it seeks out the unspoken premises of an argument and the hidden (e.g. cultural) assumptions on which a text is founded. This strategy is to show how a text relies (usually explicitly, but sometimes tacitly) on some 'binary opposition'—e.g. host/parasite, author/critic, nature/culture, melody/harmony, *bricoleur*/engineer, intercourse/masturbation—and to 'deconstruct' it, thus questioning the validity of the text or at least making its arguments more enigmatic. The reason why this strategy often has a rather damaging effect is that the opposition often has (at least implicitly) some kind of ethical force: one term is prior, privileged, 'good', and the other is derivative, 'bad'. It can be unsettling to see this picture overturned.[24]

The procedure generally goes as follows: the opposition is reduced to the primitive opposition between speech and writing discussed above. And as we saw, it was fallacious to make this division: on Derrida's analysis, the distinction breaks down when we see that all discourse (spoken or written) takes on the character of writing, and that speech itself tends to be described using metaphors that draw on the notion of writing. Drawing on the analogy, one shows that the 'good' term of the opposition—host, author, nature—cannot be described without referring to the 'bad' term, and that this interdependence eventually leads to a collapse of the distinction. This collapse will have paradoxical results: typically the text will ostensibly say one thing while, on closer examination, unwittingly suggesting the opposite.

I have neither the space nor the expertise to give examples from Derrida. I must, again, refer to the next section for some sketchy examples.

Rigour and play

A general kind of strategy that Derrida employs in his writings is to subject texts to the most meticulous examination, and to set forth arguments that conform (he claims) to the highest standards of rigour. This would not seem unusual in a British philosopher. However, set alongside the fact that these same writings of Derrida's are often lighthearted, sprinkled with wordplay and deliberately spurious reasoning, and furthermore often have no overall shape or purpose (at least not one that is at all easy to grasp), this emphasis on rigour starts to look somewhat paradoxical.

Derrida's acceptance that intellectual activity is always rooted in a given context implies that we must take that context seriously even when we are questioning it. His

[24]Incidentally, it seems to be this strategy of deconstructing binary oppositions underlying established discourses that makes Derrida attractive to the Left. Rather than storming the bastion of (e.g.) capitalism from without, the Marxist dream was to show that it must crumble from within; that capitalism contains contradictions which will lead to its own downfall. That dream evaporated when dialectical materialism was discredited. But the deconstructionist tactic of prying apart capitalist discourse by inverting and discrediting its oppositions seems to hark back to the good old days when one could make all social injustices vanish in a puff of Hegelian *Aufhebung*.

arguments are meant to show up the limits of language and reason, but to do so convincingly, they must be rigorous. This rigour is not intended to ensure *truth* in an absolute sense; rather, it is a result of following the ground rules of the context. At some later stage these same arguments may be called into question; that's not the point. The purpose of logic is simply to *allow* us to explore.

These explorations often have a rather transcendental feel to them: in particular, Derrida is fond of structuring texts around keywords like '*différance*', 'trace', 'dissemination' or 'supplement' which sum up certain aspects of writing. Within a given text he uses such a word as if, by so doing, he can 'step outside writing' and talk about it in an absolute way. But in fact, any such 'stepping outside' is always provisional, a way of luring the reader out of a context into a broader one by making the latter seem universal.[25] And indeed, Derrida rarely sticks with the same keyword for long; each new book seems to elevate a new word, and the old ones lose their portentious significance and fall into disuse or simply become convenient tags, not to be taken too seriously.[26]

This attitude to rigorous argument—accepting it and distancing oneself from it at the same time—is reflected in the overall mood of Derrida's texts. Even the ones which are most rigorous in detail are extremely lighthearted in tone: [Derrida 1972/77/88], for example, in which Derrida refuses to engage Searle in any 'serious' way but skips lightheartedly around him, darting in with passages of reasoned (and quite compelling) analysis but then stepping back before he has fully exposed himself. The effect is to distance the reader from everything: Austin's theories, Derrida's critique, Searle's reply, Derrida's riposte. We must always be knocked off balance before finding our feet; alertness to the open-ended possibilities of a text relies on our being off balance.

Wittgenstein's later writings convey the mood of a detective following a faint and confusing trail with intense concentration; by writing like this, he is *showing* us what his method entails, the fact that disentangling ourselves from a conceptual mess is a painstaking business. Similarly Derrida's destabilising combination of lightheartedness and meticulous rigour *shows* us the way in which we have to take his arguments and other people's arguments very seriously on their terms while maintaining an ironical distance from any particular form of argumentation.

This might be a good time to insert a remark about Rorty. Heavily influenced by American pragmatism and the late Wittgenstein, his position is (crudely put) that language consists of strings of noises and marks exchanged by people to achieve various ends; and that the elucidation of meaning is simply the process of correlating the noises you use with the noises I use myself. He views Derrida as above all a *writer* who shares this point of view, and whose writings convey its profound and often bizarre implications, thus charming us out of our traditional essentialist ways of thinking.

This reading seems to be a coherent one. However, it does lead Rorty to criticise Derrida's use of 'master-words' like *différance* which (Derrida claims) are not master-words and do not denote concepts. It also makes him puzzled at the fact that Derrida claims to present rigorous arguments, which Rorty often regards as 'plain awful'. I hope I have explained why I regard these criticisms as unwarranted.

---

[25] Spivak has suggested that all of Derrida's words should be regarded as being written *sous rature*, after the fashion of Heidegger; Derrida often does this in [Derrida 1967].

[26] In this regard, one might note that Derrida claims to have been unpleasantly surprised at the fate of the word 'deconstruction'.

A warning Any deconstructive reading of a text feels like it's pulling a fast one on the author. It often seems to rely on mere verbal trickery rather than engaging seriously with the text. I repeat that the 'intention' of such a reading is not to present arguments against a text on its own terms, though this process is an essential part of such a reading; in particular, the appearance of 'verbal trickery' results from attending to the actual words of the text and trying to see what they do when released from the—spurious—bonds of 'authorial intention'. I also repeat that a deconstructive reading does *not* challenge the 'validity' of a text *on its own terms*. It challenges the wider validity of that 'validity' by transfiguring, not demolishing, the text.

Some closing remarks

Before moving on to computer science, let's step back and make a few general observations. The first is that, though Derrida's thought is often claimed to be revolutionary and though his followers often regard him as an isolated genius, he is firmly positioned within some very mainstream mid-twentieth-century philosophical positions, shared by both analytic and continental philosophers. The first is the move away from subjectivism towards philosophical theories based on "intercourse and commerce between finite and historical individuals"; the second is the so-called *linguistic turn*, which sees language as the universal paradigm of all forms of human activity, such as cognitive and social processes.[27] This point of view, of course, pervades works such as [Winograd and Flores 1986].

Many of the things that Derrida says should also seem quite familiar. For instance, there are strong analogies between my 'conservative' reading of Derrida and the views put forward in [Kuhn 1962/70], especially in Section X which emphasises just how much *what we see* and how we understand the world depends on the paradigm we have adopted. A few more points of contact with Kuhn would be: a rejection of phenomenology and the idea of a universal, 'objective' language of observations,[28] a theory of meaning which incorporates Saussure's idea of meaning-as-difference and Derrida's dynamic modification of it,[29] and a denial that the philosophical ideas being put forward lead to pure relativism.[30] However, I have not seen this relationship worked out in any detail. [31]

Finally, as Derrida has trashed so many traditional philosophical views, we should reassure ourselves about how much he has preserved. As Levin points out, [32] behind all his rhetoric Derrida still adheres to the structuralist belief that meaning is not arbitrary but is contained in the relationships between terms, and that these relationships do not form a chaotic mass but have structure; the new twist on structuralism is the incorporation of incompleteness and change into the notion of structure: to invoke some Saussurean and Derridean keywords, "What différance means, then, is that structural synchrony will now, by virtue of a new definition, contain diachrony within itself, as trace." Acts of interpretation are not all *ad hoc* and disconnected; programmes are possible.

[27] See G. Márkus, "The paradigm of language: Wittgenstein, Lévi-Strauss, Gadamer", in [Fekete 1984].

[28] [Kuhn 1962/70], pp. 126–127.

[29] *Ibid*, p. 128; Derrida's point that meaning depends wholly on context is mirrored by Kuhn's remarks at p. 101.

[30] *Ibid*, p. 205.

[31] There are two allusions to Kuhn in *Derrida on the Threshold of Sense* [Llewelyn 1986], a remarkably unhelpful book which might more appropriately have been titled "Llewelyn Beyond the Threshold of Sense".

[32] C. Levin, "La Greffe du Zèle: Derrida and the cupidity of the text", in [Fekete 1984], p. 209.

Should computer scientists read Derrida?

Well, should they? I want to give some examples of what a 'deconstructionist computer science' might look like. They will be extremely sketchy, and to that extent unconvincing. And in any case one might always reply

- *This is all rubbish*; or
- *We would have thought of it anyway.*

I'm not sure what to say about the first objection. As for the second, it's the kind of thing physicists always say about mathematicians. It's one thing to have ideas; it's another thing to have a framework from which ideas emerge in some systematic way, and within which we can explore their consequences and limitations systematically.

As a warm-up let me observe that, once you've read Derrida, the phenomena he talks about seem to pop up everywhere. It is a commonplace, almost not worth remarking, that pieces of code (a shell script, a patch to an operating system, a TEX macro), once written, are often severed from their authors, used in totally unexpected contexts and modified and recombined in totally unforeseen ways. For these fragments, the notions of authorship and meaning or intention can become quite dubious. We tend to regard this notion of programming as parasitic on the standard notion—program X is written by programmer Y and marketed by company Z to solve problem W—but one can conceive of a world in which this view was inverted.[33]

Less depressingly, consider electronic mail and USENET. This takes the form of written correspondence but—probably because, for the sender, it has the immediacy and spontaneity of conversation—email messages tend to lack the kind of extra cues that letters use to try and establish an authorial presence and fix a context in which a determinate meaning can be established (the smiley convention is a notable exception). Thus email messages, particularly on emotional or contentious topics, are often peculiarly open to different interpretations. Furthermore, threads (email or particularly USENET) typically take bizarre and seemingly illogical turns. I would suggest that this is not simply because computer users are social misfits, but has something to do with the fact that, for a variety of reasons, the phenomenon of recontextualisation and the play of signifiers comes out especially vividly in electronic discourse.[34]

Am I dressing up some mundane observations in obscurantist jargon? Perhaps I am, but again, almost any kind of theorising is open to this charge. At the very least, understanding these phenomena within a theoretical framework might persuade us that they are basic things that we have to cope with and exploit, rather than irritating side-effects that we can fix, evade or ignore.

Software engineering

Derrida's position corresponds to some rather natural ideas about the nature of software engineering—perhaps giving a new perspective on some insights arrived at independently.

[33] This would arguably be a hacker's paradise and a user's inferno, but it is imaginable; and even the idea that it might be inevitable is imaginable.

[34] The logical next step, following the model of [Ronell], would be to write a book containing large passages of simulated output corrupted by random characters, explaining the role of electronic bulletin boards as a weapon of covert state terrorism. I am serious.

I will briefly deal with some general issues in software engineering and then discuss the role of formal methods.

Interaction with the user The traditional assumption is that users have certain definite requirements, and that the task of the software engineer is to discover what those requirements are and then fulfil them. In practice, as is well known, this is not at all a straightforward business. It is surprisingly difficult to come up with a precise requirements specification, and users' needs have anyway a nasty habit of changing in response to external factors or even issues that get thrown up during the software development process.

Perhaps these problems would be easier to cope with if we were to change our point of view somewhat. The trouble arises when we want to rely on the self-present voice of the user to communicate definite needs that are in some sense already 'there', waiting to be deciphered. Although we recognise that these 'needs' undergo a process of refinement and change, there is a constant urge to close off this process—to fill in all the gaps in understanding or communication, to rectify all the omissions, to reach a point where 'we now have all the necessary information'. Yet often this seems to be an impossible task; or even a meaningless one, where there is no single 'user' whose desires we are trying to fulfil, but a diverse collection of individuals within an organisation which is itself in flux.

This tension vanishes if we shift our attention away from the mythical 'user's mind' to the *dialogue* between user and software engineer. We should accept that meaning in this dialogue is always incomplete and changing, and that the process can never in principle be closed. This does not mean that we should not, at every stage, try to pin down this meaning as rigorously as possible; it only implies that we can never succeed in doing this once and for all.

This open-endedness has other implications. It means that we should no longer treat the user (if we ever did!) as an ultimate authority, nor should we believe that any inconsistency or incompleteness in a specification can be rectified solely by patient consultation with the user. 'Deconstructionist software engineering' would deploy a far wider range of analytic, literary, psychological and sociological tools, while maintaining a critical distance from any particular tool or point of view.

There is nothing new in the insight that software development is much more complex than the crude linear sequence of specification, design, implementation and testing; that there is constant feedback between stages, backtracking and interaction. And, as a result, that users cannot simply stand on the outside giving orders or expressing desires, but must be drawn into the process in a much more intimate way, while losing a certain privileged status. These are not new ideas, but perhaps having a theoretical framework within which we can understand their *inevitability* will help us to refine them further, and help them displace the older, more naive ideas about software development.

Software design methodologies Regarding pieces of code as texts, and regarding the evolution of a computing environment and the interplay between different pieces of code as a kind of discourse—and then thinking about this kind of discourse in Derridean terms—leads us to some ideas about software development which turn out to be oddly familiar.

Systems are not monolithic, designed top-down from some overall functionality requirement; they are built from autonomous processes. Software modules do not have frozen roles, but are designed to be reusable in different contexts, and possibly to play

different roles in different contexts. Modules are always underdetermined and extensible, and can always be made more specific in function when necessary. *But*, modules still have well-defined, stable descriptions within a *given* context. We could have gotten all of these ideas from Derrida, if we had not already been using them for years in object-oriented programming: see [Meyer 1988].[35]

I am not claiming that "object-orientation design is software engineering for post-structuralists". But the parallel is interesting, and worth exploring; following Derrida might lead us into more radical territory, in which the notion of 'system' is put in question, to be displaced by a fluid and open-ended notion of (constantly changing) 'environment'. In such a picture, multiple inheritance would become the norm; large stable software units would be rare, and instead we would have many smaller modules constantly combining and recombining in different ways; any overall unity would be transient, and any specification of functionality would be partial and provisional. My hesitant claim, following Derrida, is that such a picture does not imply chaos; that locally, one can still be perfectly rigorous and understand what is going on with perfect clarity.

Formal methods in software engineering Peter Naur has made the following criticism of exponents of formal methods: in their papers a formal system will be presented and analysed with the utmost rigour, yet in the conclusions there will be statements about its 'naturalness', 'convenience', 'ease of use' or 'expressiveness for practical purposes' that are not justified at all rigorously. Without going into the 'substantive' aspects of this debate (why such justification is called for, what 'rigorous' might mean, whether anecdotal evidence is admissible, and so on), let's take a closer look at the language of this exchange.

The distinction between *formal* and *informal* is accepted by both parties. Here the 'formal' consists of the mathematical machinery underlying some particular methodology: for example, first-order logic, or Martin-Löf type theory. The 'informal' is the outside world of users, programmers, practical tasks, the software engineering community. One might expect that this formal/informal distinction might parallel the primitive opposition of speech and writing. And indeed it does, but in an interesting way, or rather in two ways. For the root of the debate seems to be a disagreement (not consciously apprehended) about which way round the parallel runs.

To the formal methods expert, the formal system is speech: its symbols have a fixed and timeless meaning, and its rules are precise and unvarying. The syntactic notation expresses ideas directly and unambiguously. The outside world, with its shifting constraints, problems of interpretation, essentially imprecise requirements, has the undecidable character of writing. Certainty and rigour only make sense for a formal system.[36]

To Naur, it seems, things are the other way around. The users, the programmers, are *there* in front of each other, making their requirements and abilities known to each other directly: the discourse of the IT community has the character of speech, of interacting presences. Formal methods have the parasitic nature of writing, and it is their relationship to the real world—what the symbols have to do with the real world and how they function

[35] And of course this is not all there is to object-oriented programming. But just *what* 'object-oriented' means is, in any case, somewhat controversial.

[36] Furthermore, to someone using formal methods, the process of constructing and analysing formal specifications and proving things about them feels like a dialogue, whereas communication with users may well be entirely via documents—written texts. This is often the case when an academic department has a contract, say, to formally verify some system being used by an external organisation; only a couple of people involved with the project may have direct contact with the organisation.

in the context of the real world—which is problematic.[37]

In both cases one term is privileged over another, a 'speech' over a 'writing'. But whichever way round this is done, leads to incoherent results. The formal methodists can only apprehend the meaning of their formalisms by means of metaphors deriving from preformal practice: to the extent that the game is not completely arbitrary and meaningless, it must be derived by analogy from some preformal experience, and so the problematic nature of the informal intrudes itself at the very origin of the formal. On the other hand, Naur cannot fall back on a scientific requirement of rigour that does not, in its formulation, draw away from the immediacy of interacting presences in a community, to abstract criteria subject to the same slippages of meaning (arising from their iterability) as the symbols of a formal system.

So what's the way out? Perhaps it is to realise that the design process is a shifting discourse in which 'formal' and 'informal' methods participate on equal terms. There is a constant shuttling back and forth—for example, consider the well-known phenomenon that the process of getting a formal specification to 'work' often throws up problems on the informal level that were previously overlooked.

Different methods interact in complex and changing ways, and while it is imperative to explore the nature of this interaction in every case (so that Naur's emphasis on empirical studies is quite reasonable), and while may be convenient to draw general lessons from such investigations (not to be taken too seriously, as truly 'general'), the strict division of methods into two spheres is a pointless distraction. It draws attention away from a proper study of the unstable character of any specific language—a computer language, a specification language or a natural language—being used.

Expert systems

The implications of Derrida's ideas on the design and use of expert systems deserve a paper to themselves. Perhaps someday one will be written, by someone more qualified to write it. Meanwhile I will make only some very brief, abstract (but hopefully familiar-sounding) comments, without trying to say anything on a practical or technical level.

Firstly, the role of the human expert must be called into question. We must doubt the idea that an expert system encodes the knowledge of an expert in any direct way—and its consequence, that improving an expert system consists of refining it into a more accurate representation of the expert's knowledge. The situation is much more complex: concepts, rules and schemas, once encoded, have an independent and surprising life; they do not simply constitute an imperfect depiction of someone's mind. Like authorial intention, a human expert's knowledge is "inscribed within a system which it no longer dominates."

Secondly, it is not in principle possible to delimit precisely the domain of application of an expert system—except in the most trivial cases. There will always be marginal cases, in which adjacent (unrepresented) bodies of knowledge have a decisive influence on which decision should be taken. Furthermore, as our notion of what this domain 'is' changes, the meaning of the expert system must change. It would be irresponsible to build a system which did not take these things into account. An expert system should be able to deal with situations in which it strays out of its area of expertise, and to adapt to the

[37] Furthermore, for someone like Naur, face-to-face interaction with the user is the norm, whereas formal methods are grasped in terms of their results: the things ones sees are *Z* specifications, formal proofs, mathematical documents.

recontextualisation which accompanies its evolution.[38]

In other words, it is not enough to say that an expert system is reliable within prescribed boundaries and if we are careful we can make sure we only use it within those boundaries: within an artificially closed world. Instead, it must be able to cope with the open-ended meaning of the representations it uses. However, we should not allow ourselves to believe that such capabilities can be definitively captured in any higher-order schemas; this would only be an attempt to recapture a 'closed world' at a higher conceptual level.

Finally, our discussion implies that an expert system can never be an authority, no matter how carefully we try to delimit this authority. Rather, its operation must be seen as embedded in a complex dialogue involving many people (users, clients, managers, programmers...) and probably many other pieces of software. Any attempt to fix its role precisely must always be provisional. This does *not* mean such attempts should not be made; it does mean that they should not be taken to be definitional/definitive.

Other writers have said some of these things: see, for example, [Clancey 1989] which emphasises the importance of context and the fluid nature of representation (though the basis of the analysis, Clancey's situated perspective, is rather different). And architectures which incorporate notions of context and change from the start seem to perform much better in practice than ones which retain a flat, static picture of what an expert system should be: see [Compton et al. 1992].

Slezak on situated cognition

> I don't know *what* the constituents of a thought are, but I know *that* it must have such constituents which correspond to the words of Language... the kind of relation of the constituents of the thought and of the pictured fact is irrelevant. It would be a matter of psychology to find out... [Thoughts consist of] psychical constituents that have the same sort of relation to reality as words. What those constituents are I don't know.[39]

My last example will be a somewhat amateurish deconstruction of [Slezak 1992], a paper which puts forward a certain interpretation of (Clancey's ideas about) situated cognition.[40] Doubtless there are many 'real' arguments one could present against the views put forward there; let someone else present them. I will attempt to do something a bit different.

A warm-up: CS vs. MLT Before discussing Slezak's main argument I would like to make an apparently irrelevant aside, to do with an issue which he refers to in the closing passage of the paper.[41] This is Fodor's claim about semantics: that in explicating

[38] For example, an expert system intended to diagnose psychological disorders should not be anchored too firmly to a particular classification of disorders, a particular judgement about what conditions are in fact disorders, or a particular theory about how disorders arise. It should be able to continue operating intelligibly and making 'rational decisions' even when we change the framework in which we interpret its behaviour—within limits, of course.

[39] Wittgenstein, *Notebooks 1914–1916*; cited in [Kenny 1973], at p. 58.

[40] A 'genuine' deconstructive reading would look rather different from what I attempt: it would fasten on a small fragment of the text, subject it to a minute textual analysis, and be several hundred pages long. I am fortunately not capable of writing something like this.

[41] The following passage may be safely skipped.

the meaning of symbols in the mind, an MLT (Machine-Language with Transducers, i.e. 'transducer/procedure') approach must be parasitic on CS (Classical Semantics, i.e. model-theoretic semantics). I would like to apply a certain twist, or a form of wordplay, to this point of view.

There is an analogous situation in programming language semantics, where a given language may have both a denotational semantics (interpreting its syntax in terms of mathematical entities) and an operational semantics (describing how syntactic expressions may be evaluated to produce other syntactic expressions). The relationship between these two styles of semantics is a subtle one: one speaks of a denotational semantics being *computationally adequate* and *fully abstract* when it is 'compatible' with the operational semantics in a certain technical sense.

Different people have different views on whether denotational semantics or operational semantics is more 'fundamental'.[42] Rather than adopting one stand or the other, I want to recall a technical point: namely, that very early on, Milner proved the existence of a *unique* fully abstract model of a functional language with respect to a particular operational semantics. This denotational model is constructed *using* the operational semantics.

Conceivably, if mental representations exist, there may be analogous results about their possible semantics: given a formalised transducer/procedure semantics, we may be able to construct an 'artificial' model-theoretic semantics. But would it really be that artificial? If we were to make some kind of reductionist step (which cognitivists don't, I believe), we could argue that a faithful transducer/procedure semantics encodes our experiences and our responses, and the fact that our only notion of the world comes from our interaction with it—our only notion of 'chair', for example, consists of the way we experience and respond to chairs. On this view, the model constructed from a *completely faithful* procedural semantics *is* the 'canonical' model-theoretic semantics. And even if we reject the phenomenalism on which this argument is based, we have still shown that the 'real' model will be indistinguishable from the model we have constructed.

What was the point of this speculative (and slightly strained) argument? It is not a 'contribution' to the debate—more of an annoying interjection that is easily dismissed. Rather, it is meant to 'deconstruct' the debate by questioning the opposition on which the debate is based, drawing attention to some hidden dependencies which should perhaps be examined more closely before the debate can be resumed. In doing so it pretends to step outside the terms of the debate. This particular attempt lacks subtlety, and comes across as a little too straight-faced; but as a primitive example of 'deconstruction in conservative mode' it at least has the advantage of being brief.

Internal and external representations Back to Slezak. His essay is on 'situated cognition', a collection of loosely related positions on the nature of cognition to which he seems generally hostile. One could say that his position, cognitivism, is that there are symbols in the brain which represent things in the world, that cognition consists of manipulating these symbols according to formal rules, and that the task of AI/cognitive science is to describe these symbols and rules. All this is clearly closely related to early-Wittgensteinianism.

Situated cognition (as it appears in the quotes that Slezak has selected) rejects this picture. Cognition, rather than relying on internal representations—i.e. operating on a

[42] For example, lazy functional languages are usually understood by theoreticians in terms of their denotational semantics, while Standard ML is defined in terms of an operational semantics, and people working on it tend to rely on operational rather than denotational intuitions.

picture of the world in the brain—makes sense only within a given environment, and is the result of an unmediated interaction between our minds and the world. Knowledge, for example, is not some kind of textbook in the brain, but consists of "the potential for situated activity" (Greeno). In particular, situationists such as Clancey deny the existence of cognitive representations: knowledge, beliefs, desires and memories.[43] We are interested, here, in what Slezak makes of all this.

The core of Slezak's argument is the distinction between internal and external representations: in his view, when Clancey says that "mental representations do not exist" he really means that mental representations are a different *kind* of representation from our external representations. In Slezak's own words,

> Clancey appears to be denying [that]... internal and external representations are relevantly similar for the purposes of theoretical understanding... [His claims] can be read as reserving the term 'representation' exclusively for our own external, communicative symbols.

In my view[44] this reading puts the English language under a rather severe strain, but that is not in itself grounds for attacking it; Slezak is not butchering Clancey any more than Derrida (deliberately) butchered Searle. Rather than arguing about whether Clancey would agree with him, let's take a closer look at the implications of what he says.

The distinction that Slezak draws between internal and external representations must rest on the fact that the latter are decipherable on a conscious level and the former are not. Cf. his remark on p. 11: "Whatever may be the basis of intentionality, it cannot be the *intelligibility* of internal representations to a fully conscious mind" (made in the context of a discussion of Searle's Chinese room example). The symbols of the mind are meaningful in a *different* way, internal, autonomous way quite separate from the way in which language is meaningful: this is why "...cognition and representations could presumably develop and function in isolated, non-social organisms." (p. 13). This point is made again on p. 17:

> Once it is not required that a fully conscious intelligent system (i.e. Searle) be able to understand the meaning of [internal representations], then the entire notorious puzzle disappears.

Slezak's assertion that "there are mental representations, but they are different from external ones" embodies exactly the 'binary opposition' that we should try to deconstruct. To make the analogy with the speech/writing distinction explicit: internal and external representations are both representations, just as speech and writing are both language. But internal representations are immediate and present, like speech, while external ones are conventional and distanced, like writing. The meaning of an internal symbol is (presumably) completely specified for the mind in which it lives, existing as it does only within that mind; external symbols are conventional and underdetermined, in such a way as to make them useful for communication between people.

A budding deconstructionist might continue along the following lines: The only notion of internal representation we could possibly have comes from our notion of external, consciously graspable representation—"the privileged term ('internal representation') is

[43] See also [Winograd and Flores 1986], e.g. at p. 33.

[44] And in Slezak's: on p. 9 he admits that attributes this view to situationists 'despite their words'.

parasitic on the unprivileged". In fact the success of the strong AI project would consist of turning the former into the latter, of making internal representations intelligible and thus undermining the distinction.

On the other hand if strong AI is impossible it will be precisely because the search for mental representations is a wild goose chase; that is, because Clancey is literally right, and the assertion is false. In either case Slezak is in trouble. One can always finesse one's way out of trouble, but I hope I have made my point here.[45]

Again, I am not denying that Slezak has a plausible argument or a reasonable position. I am pointing out a certain instability in his reading of Clancey: that it relies, in an odd way, on our ignorance of whether strong AI will work or not. And it must seem strange that the validity of one position relies, not on the truth or falsity, but the *undecidability* of another.

Conclusion

I have tried to summarise some of Derrida's key ideas and attitudes, and explore some of their implications for computer science. Often deconstruction links up with ideas that are already current: it may suggest a way of structuring them within a larger framework and perhaps pushing them further, or it may offer a critique. My discussion of these implications has been unsatisfactory—often too brief, too imprecise, and too abstract. This is not simply because of lack of space; the issues have barely been explored. I firmly believe that they ought to be.

Anyone who reads Derrida expecting to find a penetrating critique of Anglo-American philosophy of mind or philosophy of language and the development of a systematic alternative within the same metaphilosophical framework, is going to be disappointed. I agree with Rorty, a self-confessed Kuhn-ite, that Derrida is important in a different sense: he is introducing us to a new, enchanting and (possibly) productive way of thinking about language, which might simply displace the old ways, rather than disproving them.

No-one can disprove that the Sun goes round the Earth, but no-one with any sense believes it anymore. Copernicus showed us to a more elegant, and ultimately more fruitful, way of understanding the universe. Deconstructionists are making similar claims for Derrida. It is hard to see how these claims could be justified, but that does not mean that his ideas are useless. I have tried to show in this paper that they lead us to some interesting reflections on a variety of different topics in computer science, some quite philosophical in nature and some quite practical. I don't think my arguments have been compelling; but I hope I have made at least a tentative case that we should be reading Derrida and reading about Derrida—not in the office, certainly, but at least in the bath.

References

[Barthes 1966] R. Barthes, *Image-Music-Text*, (tr. S. Heath), 1967.

[Bratko 1990] I. Bratko, *Prolog: Programming for Artificial Intelligence* (2nd edition), Addison-Wesley, 1990.

[45] One reply (though, I'm certain, not the strongest) might be that internal representations exist but must always remain totally inaccessible: This is definitely fishy.

[Derrida 1967] J. Derrida, *Of Grammatology* (tr. G. Ch. Spivak), Johns Hopkins University Press, 1974.

[Derrida 1972/77/88] J. Derrida, *Limited Inc*, Northwestern University Press, 1988.

[Fann 1967] K. T. Fann (ed.), *Ludwig Wittgenstein: The Man and His Philosophy*, Dell, 1967.

[Fekete 1984] J. Fekete, *The Structural Allegory: Reconstructive Encounters with the New French Thought*, Manchester University Press, 1984.

[Kenny 1973] A. Kenny, *Wittgenstein*, Pelican, 1975.

[Keynes 1936] J. M. Keynes, *The General Theory of Employment, Interest and Money*, Macmillan, 1936.

[Kuhn 1962/70] T. S. Kuhn, *The Structure of Scientific Revolutions* (Second Edition, enlarged), The University of Chicago Press, 1970.

[Llewelyn 1986] J. Llewelyn, *Derrida on the Threshold of Sense*, Macmillan, 1986.

[Meyer 1988] B. Meyer, *Object-oriented Software Construction*, Prentice-Hall, 1988.

[Norris 1982] C. Norris, *Deconstruction: Theory and Practice*, Methuen, 1982; (revised edition), Routledge, 1991.

[Norris 1987] C. Norris, *Derrida*, Fontana, 1987.

[Radford 1981] A. Radford, *Transformational Syntax*, Cambridge University Press, 1981.

[Ronell] A. Ronell, *The Telephone Book: Technology, Schizophrenia, Electric Speech.*

[Rorty 1983–90] R. Rorty, *Essays on Heidegger and Others*, Cambridge University Press, 1991.

[Thompson 1991] S. Thompson, *Type Theory and Functional Programming*, Addison-Wesley, 1991.

[Winograd and Flores 1986] T. Winograd and F. Flores, *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing, 1986.

[Winston 1981] P. H. Winston, *Artificial Intelligence*, Addison-Wesley, 1981.

[Wittgenstein 1921] L. Wittgenstein, *Tractatus Logico-Philosophicus*, (tr. Ogden and Richards) Routledge, 1922; (tr. Pears and McGuinness), 1961.

[Wittgenstein 1953] L. Wittgenstein, *Philosophical Investigations*, Basil Blackwell, 1953; (second edition), 1958.

[Wood 1990] D. Wood, *Philosophy at the Limit*, Unwin Hyman, 1990.

[Clancey 1989] W. J. Clancey, "The frame of reference problem in the design of intelligent machines", in K. van Lehn and A. Newell (eds.), *Architectures for Intelligence: the Twenty-Second Carnegie-Mellon Symposium on Cognition*, Hillsdale: Lawrence Erlbaum Associates.

[Compton et al. 1992] P. Compton, G. Edwards, B. Kang, L. Lazarus, R. Malor, P. Preston and A. Srinivasan, "Ripple down rules: turning knowledge acquisition into knowledge maintenance", *Artificial Intelligence in Medicine* 4 (1992) 463–475.

[Compton and Jansen 1990] P. Compton and R. Jansen, "A philosophical basis for knowledge acquisition", *Knowledge acquisition* (1990) 2, 241–257.

[Gaines 1991] B. R. Gaines, "Between neuron, culture and logic: explicating the cognitive nexus", *ICO: Intelligence Artificiel et Sciences Cognitives au Quebec*, Vol. 3 (2) Été 1991, 47–61.

[Gaines and Shaw 1990] B. R. Gaines, M. L. G. Shaw, "Cognitive and logical foundations of knowledge acquisition", 5th AAAI-Sponsored Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada, November 1990.

[Slezak 1992] P. Slezak, "Situated cognition: minds-in-machines or friendly photocopiers?", invited talk at the McDonnell Foundation Conference on Cognitive Science, Santa Fe, New Mexico, June 1992; and at the Second Australasian Conference on Cognitive Science, University of Melbourne, February 1993.

[Wilson 1992] E. Wilson, "Psychology and the scene of writing: Freud, Derrida and connectionist theories of cognition", *Not My Department: Papers from the Seminar Series, Volume Two*, Not My Department Publications, 1992.