Two Lessons of Logic[†]

Brian Cantwell Smith[*]
University of Toronto

1 Background

Modem logic is most reasonably dated to 1879.[1] This means that logicians have had more than a century to study a particular family of so-called formal systems. Not surprisingly, much of the enormous amount they have learned is peculiar to their specific assumptions about the foundations of mathematics. Some of their insights, however, are universal, holding for any system of signs, symbols, or sentences.

McDermott has come to see that many of logic's particular assumptions are not applicable to general human reasoning. By and large 1 think he is right, and I agree that the consequences are a little appalling. But McDermott goes on to imply that we must therefore reject logic as a whole, at least as a basis for artificial in-

---

[†]Original published in Computational Intelligence, 3(33), 214–18 (1987), in response to Drew McDermott, "A critique of pure reason." *Computational Intelligence* 3(33), pp. 151–60, 1987.

[*]Coach House Institute, Faculty of Information, University of Toronto
90 Wellesley St W, Toronto, Ontario M5S 1C5 Canada

Last edited: November 14, 2009
Comments welcome

brian.cantwell.smith@utoronto.ca

[1]The year 1879 was the publication date of Frege's *Begriffsschrift*, the first step in his life-long attempt to develop a logical foundation for mathematics. Although I will argue that we should retain the rigour and precision of the enormous tradition that Frege's work inspired, I believe that the intuitions and insights of various earlier writers on symbolic systems, especially Charles Sanders Peirce, are at least as relevant to AI's concerns. From our point of view it is unfortunate that the vast majority of logical development, during this century, has been devoted to the narrower, purely "mathematical," case.

telligence (AI). It is time, he suggests, to end the love affair between AI and logic.

Now I do not care too much about the term "logic"—whether we should use it, broadly construed, for the full range of rational belief revision (roughly, what you should or are likely to believe next, if you believe *P* now), or whether we should follow traditional mathematical logic in confining it to the entailment relation (roughly, what follows, if *P* is true), and adopt a more general term for the human case. (As I say, I do not care, in some ultimate sense, but in what follows I will use "logic" for the narrow, entailment sense, and "thought" for what we do.)

What I do care about is this: that we learn everything we can from those hundred years of intellectual history. More specifically, I worry that McDermott, in rejecting logic's particular assumptions, is also discarding some of its universal lessons. Two lessons, in particular.

1.1 Lesson one: the irreducibility of content to form

The first lesson I take to be the deepest truth that logicians have uncovered: that there is more to a symbol system than can be gleaned from its rules and representations. In particular cases this can be made quite concrete (incompleteness results for arithmetic, for example), but the lesson itself is general. For discussion, I will call it the *irreducibility of content to form*. To get at it, we need to distinguish two somewhat independent aspects or dimensions of any symbol system.

What I will call the **first factor** of a symbol system involves the shapes of the symbols, the ways they can be put together and taken apart, and the behaviour or operations defined over them. Expressions, predicate letters, and modus ponens in logic; abstract data types and corresponding operations in computer science—that sort of thing. You can think of this whole package as a representational system's *mechanics*: a combination of its static structures and dynamic operations defined over them. The first factor is also what must be directly realized in a physical substrate, if the system is going to do any work.

The **second factor** has to do with what the symbols *mean*, what they are about—their *content*. Interpretation (in the logician's sense!) is a second factor phenomenon, as are truth and ref-

erence, the latter in the sense of the relation between the name 'McDermott' and a person who works at Yale. Shades of the old declarative/procedural distinction (but only shades; see Smith 1987).

In mathematical logic the two stories are called *proof theory* and *model theory*, respectively; semantics is taken to be the study of the latter. Furthermore, semantics—the story about content—is what really matters about these systems; in this sense it is "more equal" than the first factor account. There is a reason for this asymmetry: without some second factor aspects you could not be sure you have a symbol system at all. *Everything* has a mechanical nature, to put this another way; it is metaphysically prior (that is why I called it first). Having content, on the other hand, and therefore being amenable to a second factor analysis, is what distinguishes symbolic or representational (what philosophers call intentional) systems like languages and computers from ordinary physical objects like hockey pucks and oil refineries.

The first lesson of logic, then, can be stated in terms of the "second factor": the content of a symbol, at least in general, is not an intrinsic or causally proximate property of it, but arises as a relation between the system and some other domain. For example, the meaning or reference of the symbol PLANE$_1$, in an axiomatization of this morning's air traffic over LaGuardia. would involve some actual airplane, two thousand feet up. No amount of investigating how the symbol PLANE$_1$ is used within the air traffic control system could ever tell you what plane it refers to. To get to the plane itself you would have to look outside the system, to see how it was connected up to, and used in, its environment. Or imagine trying to determine the truth of a report claiming that 70% of all doctors recommend Crest toothpaste. You would not bring out your microscope to study the paper that the report was written on, or submit it for typographic analysis; you would drive around and actually talk to doctors.

Content, in other words, does not hang around a symbol system like a nervous teenager afraid to leave the house; it is out there, in the world. Nor can the second factor be deduced from the first. Furthermore (this is perhaps the most surprising thing of all) this externality of content does not arise only for realtime systems, like air-traffic control systems. It holds even for as abstract, circum-

stantially independent, causally inert, and completely disembodied a system as formal arithmetic.

It is a major corollary to this first lesson that content relations are not computed. If I use the name 'Povungnituk' to refer to a small town on Hudson's Bay, for example, a content relation holds between my utterance and a major source of Inuit stone carvings. But, just like the property of being the average age in a collection of people, this relation just *is*; no work needs to be done in order for it to hold. Admittedly, in interpreting my utterance, you may "compute" something, but the purpose of your computation will only be to arrange yourself to stand in something like the same kind of (non-computed) content relation to the town that I did when I said its name. The town itself, which is a part of the content relation, is not a part—not what the philosophers call a "proximate" cause—of any computational activity of saying or understanding.

In contrast with the first factor, to put this another way, the second factor of a symbol system does not need direct physical realization. There is no way the Inuit could deploy a sensor in Povungnituk to detect whether their town was being referred to by an arbitrary speaker in an arbitrarily remote location.

It should be admitted that how this all works—how symbols "reach out and touch someone," to use AT&T's phrase—remains an almost total mystery. Some people (Winograd, for example) argue that they do only through human use; others (I am in this category) believe that human interpretation is sufficient, but not necessary.[a] But whatever one's view, the facts that these views have to deal with are impressive. To start with, reference outstrips causality, at least locally; with four simple letters I can refer to a composer who has not existed for more than 200 years—not to a set-theoretic model of him, to his name, or to your comprehension of him, but to his very heart and soul (even though, unlike Povungnituk, he does not exist any more). Reference relations are not even constrained by the light-cone; as Church once put it, "semantics travels at the 'speed of logic'."[2] In fact reference almost outstrips comprehension; if we did not know that language works, we

[a] «Yikes; figure out what to say, here»
[2] «Ref CSLI seminar, May 3, 1984».

would spurn rumours of its long distance capabilities.

What is more, the little we do know is not reassuring—at least for AI. To take just one example, it seems that the axioms of arithmetic must be connected to the numbers (rather than to other non-standard interpretations) not by anything intrinsic to them, but by us humans. It seems, that is, that agents are what matter, for semantical connection. But that only shifts the mystery—squarely onto AI's subject matter: cognitive agents, interpreters (again, in the philosophical sense!) of the symbols and signs.

But if we d not know *how* reference and content work, at least we know *that* they work, and that there is more to it all than proof theory. Furthermore, I take it to be the job of semantics, at least as classically understood, to explain, in as systematic and rigorous a way as possible, the interplay between "formal" (i.e., first factor') properties, on the one hand, and these relatively more mysterious second-factor relations of meaning and content, on the other. Admittedly, in the face of considerable ignorance, we cannot yet fill in all the details (though we do what we can—which, to date, mostly means enumerating the relata, but someday more should be possible). What matters—what logic's first lesson really tells us—is that the second factor content story must be told.

1.2 Lesson two: a single theoretical stance

Logic's second lesson is a theoretical one, in the sense of being about theory—about how symbolic systems should be explained. Related to the first, it arises from the recognition that the two factors (proof theory and model theory, in the traditional case) must be related, in spite of being conceptually distinct. This is the role, in logic's case, played by completeness proofs, notions of soundness and validity, etc. In contrast, you could also argue that there are two stories to be told about money: one about its physical embodiment, one about its social and economic import. But, at least on the surface, there is no obvious reason why the stories should relate; engineers at the Franklin Mint designing new dollar bills probably do not need to know Gresham's law (that bad money drives out good). In logic, however, the connection is more direct. You could not really claim to have a (first-factor) inference regimen if you could not relate it pretty directly to (second-factor) semantic interpretation.

Because of this global but crucial connection, logicians have developed a single theoretical stance from which to tell both stories. The stories, furthermore, overlap in vocabulary: the same theoretician's grammar that spells out the linguistic regularities of logical formulae is used by proof theoretician and model theoretician alike. And the overlap is necessary. That logic's two factors are relatively independent (more on this in a moment), and yet must ultimately be related, can only be said from a vantage point from which they can both be seen.

In laying these things out it is important not to confuse the conceptual distinctness of factors, or the singleness of theoretical standpoint, with the question of how related the two factors are. By analogy, geometry distinguishes length and area, but then goes on to tie them strongly together, in the familiar way. Similarly, both classical and relativistic mechanics view time and space as conceptually distinct; the two theories differ in what they then say about the notions—whether they are independent (the classical case) or intimately related (relativistic). In the next section I will claim that the first and second factors of thought should be intimately and constantly related, but it does not follow that I think they are the same thing.

How, then, does the second lesson relate to the first? Because we are now talking about theories of logic, not just about logic itself, things get a little bit complicated. In particular, since these theories (like the logical systems they are about) are themselves intentional phenomena, we have two symbol systems to consider, not just one. Lots of people have wrestled with how they relate: from Tarski, in setting up preconditions on satisfying convention T, to Quine, worrying about the radical indeterminacy of translation. But the overall structure of the connection is clear enough. *The second factor content of the theoretical accounr* (for example, the content of Kripke's 1963 paper[3]) *must include the complete first and second factor dimensions of the system under investigation* (the syntax. proof relations, and model structures of modal logic, in Kripke's case). That is just what it is to say that the theory is *about* the system under investigation.

Enough intricacies. What matters here is that a single, unified

[3]«Ref»

theory must provide accounts of *both* factors. I take this recognition of the need for a single theoretical vantage point to be the second of logic's great contributions.

So much for background. Let's turn to McDermott.

2 Human cognition and logic's assumptions

I have already indicated that I agree with much of what McDermott says: that rationality is not pure deduction over passive logical formulae, that use is an inextricable aspect of knowledge, all that stuff. But let us go a little slowly, to see just where these agreements lead. In particular, let us go back to a bit in history.

As suggested at the outset, the originators of modem so-called "formal" logic—Frege, Russell, Whitehead, Carnap, and so on—were primarily exploring (or at least motivated by) issues in the foundations of mathematics. All things considered, they did an excellent job.

Unfortunately, they also died. We, their descendents, have been so impressed by their achievement that we are in danger of thinking that they *defined what semantics must be like*. This raises a two-stage problem, related to the question raised at the outset about the relation between particular and universal insights. At first blush, we are liable to accept their proposals too glibly, not having them around to tell us why they made the decisions they did. Then, once we discover that their particular choices are untenable for our purposes, we are in danger of throwing the whole thing away, baby *cum* bathwater.[b]

Rather than trying to canvass all the choices that were made, let me simply list three assumptions underlying traditional formal logic that I believe are untenable for *AI*. The following, in other words, are tenets we must *reject:*

1. That *use* can be ignored. This premise leads logic to ignore agents and processing, to set aside context, and to focus on sentence *types* instead of *tokens* or *individual utterances*. It also suggests that a sentence must represent its whole content explicitly, since no other resources are licensed that could make other contributions. This is quite different

[b] «Point forwards to *Rehabilitating Representation*»

from natural language, where dynamic and contextual factors often implicitly contribute to the content (the time of utterance, for example, provides an interpretation for the word 'now').

2. That locally the two symbolic factors can be treated independently, even though (as suggested above) they must ultimately be globally related. In particular, proof theory or inference (first factor) and model theory or semantics (second factor) are tied together, in traditional logic, for any given system, only "at the end," with soundness and completeness theorems. From step to step, in a "formal" proof, the (first-factor) inference procedure cannot depend on or affect (second-factor) semantic interpretation. (In fact this is what "formal" is taken to mean, by theorists as diverse as Fodor and Martin-Löf.[4])

3. That language and modelling (two species of representation, I take it) should be treated completely differently. The linguistic *reference* relation—the primary subject matter—is assumed to be strictly non-transitive, engendering such familiar constructs as the use/mention distinction, hierarchies of metalanguages, and convention T. *Modelling*, on the other hand (of the sort that treats Turing machines as sets of quadruples, Truth and Falsity as 0 and 1. and so forth) is not only taken to be transitive, but also to be "free," in the sense that you are allowed to use a model of *X* in place of *X* itself (even to identify the two) with theoretical abandon.

I do not know exactly what McDermott means by "Tarskian semantics," since he clearly intends it to be broad enough to include denotational analyses of programming languages (on which more below), but I take it to mean roughly a semantical account that adopts all three of these assumptions. At any rate I will use that definition here.

McDermott's position can now be stated in terms of the first two assumptions: he recognizes that logic makes them, that AI must re-

[4] «Refs»

ject them, and that the theoretical consequences of this rejection are daunting.

I agree. I also have real sympathy for the strength of his reaction: none of these assumptions can easily be "let go of" or altered, as if that were a minor adjustment to what remains basically a purely "logical" enterprise. These are foundational assumptions, with all that that implies.

To make that concrete, let me say a few more words about each. To start with, the central intuition underlying the so-called "situated" language and computation project at the Center for the Study of Language and Information (CSLI) at Stanford involves replacing the first assumption with its exact opposite: a committed and direct focus on language use. The goal is to develop new theories and semantical frameworks that analyze individual utterances, and to embrace the crucial role of circumstance and context. To take just one example, this involves diagnosing the relevant structure of all pertinent contextual factors: relevant background facts (such as the place where "It is 4:00 p.m." was said), presuppositions, discourse structure (that help resolve pronouns, for example), facts about the language being used, the structure of the subject matter or described situation with respect to which linguistic and cognitive processes can in turn be structured, mutual belief structures that explain what can and cannot be said, internal facts about cognitive architecture that pertain to the interpretation of internal structures, and so on and so forth. And this is just one set of issues that have to be considered. CSLI has had dozens of people working on this project for four years so far—and, from my biased perspective, I think it is making good progress. In not more than another four years there should be something substantial to show.[c]

The second assumption—that first and second factors are locally independent—goes just as deep; I also think this is the one that has so sobered McDermott. He claims that the culprit of logicism is its notion of deduction (inference only to provable consequences), but if a different semantical connection (say, abduction) were semantically definable in such a way that the procedural role

[c] «Comment on how naïve that claim was…and say something about what happened.»

(first factor) could be cleft from semantical import (second factor), then it would still make sense to write things down first, and build programs second—the putative essence of the logicist enterprise. But these are subtleties: logic does assume local independence [between the two factors], and I do not believe thought is like that.

My own strategy, in attacking this one, has been to use computationally-internal notions of reflection and introspection as a crucible in which to work out viable alternatives. The point of 2-Lisp,[5] in particular, was to show that even warhorse programming languages are best understood in terms of locally intertwined factors. As it happens, 2-Lisp ignored contextual aspects of use (by design)—thereby drawing something of a distinction between the first and second assumptions—but at the same time making its architecture less generalizable than one might like. I hardly need add that a great deal more work is necessary here.

Similarly the third—that language and modelling are categorically distinct. Although McDermott does not address this premise explicitly, it stands in equal need of reconstruction. Whole new theories of representation and correspondence will be required.[6] There are two reasons this revamping is urgent: partly to explain computational practice (see below), and partly to clarify our standard theoretical apparatus. In particular, although promiscuous modelling may be helpful in answering large-scale and hence rather coarse-grained questions (such as whether a given formula is true, decidable, computable), it can be pernicious when one asks fine-grained questions about control, intensional identity, and the use of finite resources. Also, current computational systems involve representational structures of all kinds, ranging continuously from linguistic expressions to virtually iconic isomorphisms like bit maps and simulation structures. This is a large area where the particular assumptions of mathematical logic have led to untenable methodological practices (for AI theorists), as well as to untenable claims on our primary subject matter.

All in all, in other words, I agree with McDermott that the consequences of rejecting these assumptions are enormous. And yes, they certainly undermine the coherence of the "logicist" program.

---

[5]Smith (1984).

[6]See Smith (1987) for some initial analyses.

Writing knowledge down in advance, without regard to use, is a conceptual error doomed to failure.

3 What then?

But—and this is really where I have been driving—what are we to do instead? Instead of abandoning hope and reverting to unconstrained symbol mongering, surely the task is to develop alternative theoretical frameworks.

Now McDermott does not really argue for pure symbol mongering, but he does suggest that the "proceduralist" paradigm is the only other game in town, insinuating that there could not be any others. Why should that be true? For example, why should we not develop a full-scale theory of use—flesh out the project that the philosophy of science has only just started, for example—and uncover the regularities that must underlie integrated content and behaviour? From the fact that use and content are inextricably linked it does not follow that rationality is random. And if it is not random, we can understand it (at least that seems like a plausible intellectual creed).

See, this is really what I think has "got McDermott's goat." Computational practice—what our programs actually do, not what we *say* about them—does not honour logic's three assumptions laid out above; it mixes behaviour and content as richly and thickly as we do. The only rigorous semantical theories we have, on the other hand, do make those restrictive assumptions. McDermott, [quite properly and insightfully,] sees that [logic's] assumptions are untenable, and correctly notes that there are not any other proposals around ("…it must have a semantics; so it must have a Tarskian semantics, because there is no other candidate"). *So* he is forced to laud practice. But then in virtually the same breath he admits that that makes him uncomfortable. So he ends up somewhat confused.

What he *should* endorse (I claim) is not practice itself, but *theoretical frameworks that do justice to that practice.* That is, what we want is a conceptual backdrop in terms of which to understand Forbus' work in the same way that logic and model theory form a conceptual backdrop for Hayes' research on the ontology of liquids.[7] This situation is pictured in figure 1. To be fair, McDermott says a lot of things suggesting that he agrees with the general thrust of this diagram: "there are large classes of programs that lack any kind of theoretical underpinnings," "AI programs are notorious for being impenetrably complex…but a model that we don't understand is not a model at all," "what's really bothering me is that these (diagnosis) program embody tacit theories of abduction," etc. What he does not suggest, at least sufficiently explicitly, is that we need theories to do justice to programs in just the way that logic provides theories that do justice to (mathematical) sentences.

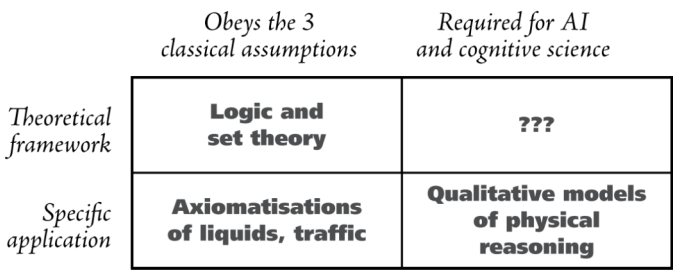

Figure 1 — Appropriate theoretical frameworks

The question, that is, is how we are going to fill in the missing quadrant. It seems that there are two evident suggestions. We could take logic and set theory, and try to modify them. Or we could throw away logic and set theory, and simply study the practice itself, like entomologists studying bees.

With respect to modifying logic and set theory, I have already said a little about what I think would be required (build in the opposite of the three assumptions listed above). And I have said it will be hard. I agree with Israel (and, I take it, McDermott) that incremental variants like non-monotonic logic are nothing like strong enough.[8] The problem is that once you start revamping this much of logic's foundations, it is not clear what remains. It is easy to say that one must understand just what aspects of classical logic

[7] «Refs»
[8] Israel (1980).

are particular (i.e., specific to metamathematics), which are universal—but that does not make it easy to do. So far I have suggested only two lessons that I believe we should view as universal, and hence as relevant to the AI case: external (non-computed!) content, and a single theoretical vantage point. But there is a lot more work to do.

I also have great respect for the other suggestion: studying and reconstructing practice. We should certainly understand architecture, physical embodiment, resource allocation—all the usual stuff. In fact this is where most of my own work has concentrated. But it is also where my original worry comes back to roost: the worry that logic's two great lessons will be lost. In fact this worry can be seen as a triple threat.

First, because it potentially confuses practice itself with theories that do justice to such practice, I am afraid that McDermott's paper will lead people to discard logic's theoretical stance completely, and focus too much on the practical side. To be honest, I do not expect McDermott himself to do this (he is too unremittingly theoretical), but a casual reader could easily mistake his intentions.

Second, if you just look at programs, and try to make sense out of what they are doing, you will be liable to focus solely on first-factor aspects of systems, for a simple reason: The first factor, as we said above, is the one that needs to be realized in the machine. And since the point of programs is to conjure up an otherwise unorganized state machine into appropriate form to exhibit reasoning, the program only needs to concentrate on first-factor problems: structures, operations, behaviour. In general, as we saw at the outset, *the content is not in the machine at all.*

The third problem arises from a curious fact about how practice is currently understood. It is hard to tell exactly what McDermott means, but the words 'program' and 'denotational semantics,' when uttered in one sentence, inevitably bring to mind the denotational semantics tradition in computer science, as illustrated by Gordon, Plotkin, etc.[9] Now I firmly believe that all current computational systems—from Amord to Zetalisp[10]—blend both factors we talked about earlier. The only factor of computational systems

[9]See for example Gordon (1979).
[10]«Refs»

that computer science talks about, however, is the first: procedural role. I believe this is true, curiously enough, *even when people use the term 'semantics.'*

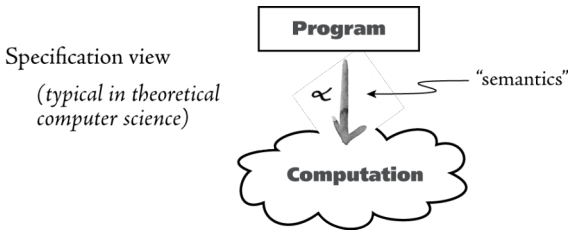

Figure 2 — The computer science view of programs

What is *called* "denotational semantics" in computer science is in fact *a model-theoretic analysis of first-factor procedural role*, for a reason that depends on an ambiguity in the use of the word 'program.'[xxx] As discussed in Smith (1987), in particular, theoretical computer science by and large views programs **specifications** of computational behaviour, as suggested in figure 2. In AI, however, it is more common—and McDermott is clearly of this view—to take them to be **constituents** within computations, as suggested in figure 3. On the specificational view espoused in computer science, as indicated in the diagram, the semantic content of a program is taken to be *the computation specified*; hence, the computation itself is what denotational semantics takes its subject matter to be. In AI, in contrast, on the ingredient view, the semantics of the program lie in the external task domain that the "AI program" (i.e,. the compu-

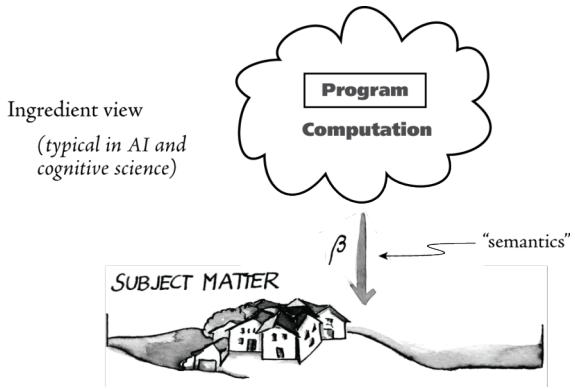

Figure 3 — Programs in AI and cognitive science

[xxx]From here through the end of this section the text has been mildly rewritten, for this version, to increase clarity. The introduction of a section boundary for a conclusion (#4) is also new; though its content—the final two paragraphs of the paper—are as in the original. See also "One Hundred Billion Lines of C++."»

tation that results from running it) is reasoning about.

One can speculate as to why the two readings have developed—perhaps because computer science more typically deals with languages which are compiled; AI programs are often (especially initially) written in languages such as Lisp, which are often "interpreted." Whatever; the point is only that clarity in regards to how programs are viewed is clearly a prerequisite to semantical clarity (and unclarity, correlatively, is a reliable source of confusion). It should also be noted that this diagnosis of ambiguity renders intelligible many computer science practices that can otherwise seem odd or even inexplicable to theorists in AI, cognitive science, and/or philosophy of mind:

1. The fact that computer science traditionally assumes a specificational view of programs explains why programming language theorists so often use term models, initial algebras, and other such constructions in their semantic analyses; they need to individuate their semantical models (remember the third assumption!) extremely finely, because what they are really modeling is (the behaviour of) *the computational processes themselves*.

2. Since it is universally assumed, in both computer science and AI, that one be able to produce or instantiate the computation associated with a program (whatever one takes the relation between them to be), programming language theorists and other computer scientists tend to believe that semantic relations must be constrained to be **effective**—in a way that semantic relations for natural language, "mentalese," and AI clearly are not (remember the first lesson!). Programs, to use Nygaard's phrase, are "prescriptions" as well as "descriptions."[11]

3. The analysis makes it clear why, from the point of view of theoretical computer science, operational and denotational semantics are taken to be *two different kinds of analysis of the same relation*—and why it is standard to see equivalence proofs between them. Computer science's distinction between operational and denotational semantics, in other

[11] «Ref»

words, is *fundamentally different* from the one we have been talking about between first and second factors—e.g., between proof theory and model theory in traditional logic. (While it is perfectly standard to prove *completeness*, it would at least be conceptually perverse, are more likely represent an outright misunderstanding, to attempt to prove that inference and entailment were *equivalent*.)

4. Finally, note that, under the specificational view, a computer really *does* interpret a program, in the logical or philosophical sense![zzz]

But enough about other people's worries. The important point here, with respect to the place of logic in Artificial Intelligence, is that the content relation that AI needs to study, as opposed to that in which theoretical computer science is interested, is the one that holds between the computational process and the world outside it—i.e., the one labeled β in figure 3. If one adopts an "ingredient" view of programs, this is just the semantics of the program itself; and so talk about *the semantics of the program* and *the semantics of the computation* amount to essentially the same thing. From the specificational perspective adopted in computer science, in contrast, the only way in which to name or identify the content relation in which AI is interested is with the phrase "*the semantics of the semantics of the program*"—i.e., as something *two semantic levels away* from the program itself.[yyy]

What ultimately matters, however, is the nature of relation β, not how one refers to it. And in this regard, three things must always be kept in mind: (i) at least in general, it will reach outside the machine; (ii) it will not (again, in general) be effective; and (iii) it will never be computed.

4 Conclusion

Return, finally, to McDermott. We had noted that a thoroughgoing reconstruction from first principles was an enormous theoretical task, and were looking at the other way of proceeding—by reconstructing practice. With respect to the latter alternative, I had

[zzz] This in spite of the ironic fact that it is more common in AI than in other parts of computer science to use "interpreted" languages.

[yyy] «Refer to the conversation with Plotkin, in which he smiled—but find out where else I said that.»

constructing practice. With respect to the latter alternative, I had three worries. First, if (as a casual reader of McDermott) you merely endorse practice, you are liable to remain [fatally] atheoretical. Second, if you try to reconstruct practice *de novo*, you are not only faced with an enormous task, but are liable to see only first-factor aspects, since those are the only ones that are implemented (content, remember, [to repeat this point one final time,] does not appear in the program at all). And then third, the twister: if you borrow techniques from theoretical computer science, you will [find yourself using semantical vocabulary, but in spite of that fact focusing] on the wrong relation completely. Furthermore, not only does [such an approach] fail to focus on the semantical relation we are interested in; for somewhat gratuitous reasons (the fact that programs are prescriptive), it also ignores logic's first lesson: the irreducibility of content to form.

No matter how you do it, in other words, there's a danger that you will miss out on logic's two great lessons. And that—I hope McDermott will agree—would be tragic.

References

Gordon, Michael (1979). *The Denotational Description of Programming Languages: An Introduction*. New York: Springer-Verlag.

Israel, David (1980). What's Wrong with Non-Monotonic Logic? *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford, CA, pp. 99–101.

Kripke, Saul (1963). Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16: 83–94.

Smith, Brian Cantwell (1984). Reflection and Semantics in Lisp. *Conference Record of the Eleventh Annual ACM Symposium on Principles of Programming Languages*, Salt Lake City, UT, pp. 23-35. Also available as Xerox Palo Alto Research Center Intelligent Systems Laboratory Technical Report ISL-5, Palo Alto, CA, 1984.

Smith, Brian Cantwell (1986). Varieties of Self-Reference. In Joseph Halpern (ed.), *Theoretical Aspects of Reasoning about Knowledge*. Los Altos, CA: Morgan Kaufman, pp. 19-43.

Smith, Brian Cantwell (1987). The Correspondence Continuum. CSLI Technical Report CSLI-87-71, Stanford University, Stanford, CA.