ceedings of the 7th International Joint Conference on Artificial Intelligence, Vancouver, B.C., pp. 331–337.

——— 1983. Natural language processing: crucible for computational theories of cognition. Proceedings of the 8th International Joint Conference on Artificial Intelligence, Karlsruhe. West Germany, pp. 1180–1186.

SLAGLE, J. R. 1971. Artificial intelligence: the heuristic programming approach. McGraw-Hill, New York, NY.

Two lessons of logic

BRIAN CANTWELL SMITH
Xerox Palo Alto Research Center, Palo Alto, CA 94304, U.S.A.
and
Center for the Study of Language and Information, Stanford University, Stanford, CA 94305, U.S.A.

Comput. Intell. 3, 214–218 (1987)

1. Background

Modern 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 aren't applicable to general human reasoning. By and large I think he's 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 AI. It's time, he suggests, to end the love affair between AI and logic.

Now I don't 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 don't care, in some ultimate sense, but in what follows I'll 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 100 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's 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'll 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'll 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're about — their content. Interpretation (in the logician's sense!) is a second factor phenomenon, as are truth and reference, 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's "more equal" than the first factor account. There's a reason for this asymmetry: Without some second factor aspects you couldn't be sure you have a symbol system at all. Everything has a mechanical nature, to put this another way; it's metaphysically prior (that's 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 hocky 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, isn't 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_{17}, in an axiomatization of this morning's air traffic over LaGuardia, would involve some actual airplane, 2000 ft up. No amount of

[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's unfortunate that the vast majority of mathematical development, during this century, has been devoted to the narrower, purely "mathematical," case.

investigating how the symbol PLANE$_{17}$ is used within the air-traffic control system could ever tell you what plane it refers to. To get to the plane itself you'd 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 wouldn't bring out your microscope to study the paper the report was written on, or submit it for typographic analysis; you would drive around and actually talk to doctors.

Content, in other words, doesn't hang around a symbol system, like a nervous teenager afraid to leave the house; it's 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 doesn't arise only for real-time systems, like air-traffic control systems. It holds even for as abstract, circumstantially independent, causally inert, and completely disembodied a system as formal arithmetic.

It's a major corollary to this first lesson that content relations aren't 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 it. 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 doesn't need direct physical realization. There is no way the Inuits 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" — remains an almost total mystery. Some people (Winograd, for example) argue that they only do through human use; others (I'm in this category) believe that human interpretation is sufficient, but not necessary. 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 hasn't 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 doesn't exist any more). Reference relations aren't even constrained by the light-cone; as Church once put it,[2] semantics travels at the "speed of logic." In fact reference almost outstrips comprehension; if we didn't know that language works, we would spurn rumours of its long-distance capabilities.

What's more, the little we do know isn't 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 nonstandard 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 don't 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[3]) 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 can't 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 don't need to know Gresham's law (that bad money drives out good). In logic, however, the connection is more direct. You couldn't really claim to have a (first-factor) inference regimen if you couldn't 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 doesn't follow that I think they are the same thing.

How, then, does the second lesson relate to the first? Because we're 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

[2]CSLI seminar, May 3, 1984.

[3]The claim that formality, in the end, reduces to first factor notions of physical realizability is argued in a book in preparation by B. C. Smith, entitled *Is Computation Formal?* (To be published by The MIT Press/A Bradford Book, Cambridge, MA.)

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 account* (for example, the content of Kripke's 1963 paper) *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's 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 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've already indicated that I agree with much of what McDermott says: that rationality isn't pure deduction over passive logical formulae, that use is an inextricable aspect of knowledge, all that stuff. But let's go a little slowly, to see just where these agreements lead. In particular, let's go back to a bit in history.

As suggested at the outset, the originators of modern 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, though, they also died. We, their descendents, have been so impressed by their achievement that we're 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're in danger of throwing the whole thing away, baby *cum* bathwater.

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 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, 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-Lof.)

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 nontransitive, 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 "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 don't 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'll 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 reject them, and that the theoretical consequences of this rejection are daunting.

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

To make that concrete, a few more words about each. To start with, the central intuition underlying the so-called "situated" language and computation project at CSLI 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 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's 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 can't 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 4 years so far — and, from my biased perspective, I think it's making good progress. In not more than another 4 years there should be something substantial to show.

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 defineable in such a way that the procedural role (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, and I don't 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 cible in which to work out viable alternatives. The point of 2-Lisp, Smith (1984) in particular, was to show that even war-horse 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 lots more work is necessary here.

Similarly the third — that language and modelling are categorically distinct. Although McDermott doesn't address this premise explicitly, it stands in equal need of reconstruction. Whole new theories of representation and correspondence will be required (see Smith 1987 for some initial analyses). 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. 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've 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 doesn't really argue for pure symbol mongering, but he does suggest that the "proceduralist" paradigm is the only other game in town, insinuating that there couldn't be any others. Why should that be true? For example, why shouldn't we 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 doesn't follow that rationality is random. And if it isn't random, we can understand it (at least that seems like a plausible intellectual creed).

See, this is really what I think's got McDermott's goat. Computational practice — what our programs actually do, not what we *say* about them — doesn't 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 sees that the assumptions are untenable, and correctly notes that there aren't any other proposals around ("... it must have a semantics; so it must have a Tarskian semantics, because there is no other candidate"). So he's forced to laud practice. But then in virtually the same breath he admits that that makes him uncomfortable. So he ends up somewhat confused.

	Obeys three classical assumptions	Required for AI & cognitive science
Theoretical framework	Logic and set theory	???
Specific application	Axiomatizations of liquids, traffic, ...	Qualitative models of physical reasoning

FIG. 1. Appropriate theoretical frameworks.

What he should endorse (it says here) isn't 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's work in the same way that logic and model theory form a conceptual backdrop for Hayes' research on the ontology of liquids. This situation is pictured in Fig. 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 doesn't 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.

The question, that is, is how we are going to fill in the missing quadrant. It seems that there are two obvious 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've already said a little about what I think would be required (build in the opposite of the three assumptions listed above). And I've said it will be hard. I agree with Israel (and, I take it, McDermott) that incremental variants like nonmonotic logic are nothing like strong enough (Israel 1980). The problem is that once you start revamping this much of logic's foundations, it's not clear what remains. It's easy to say that one must understand just what aspects of classical logic are particular (i.e., specific to metamathematics), which are universal — but that doesn't make it easy to do. So far I've only suggested two universal lessons: external (noncomputed!) content, and a single theoretical vantage point. But there's 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'm 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 don't think McDermott himself will do this (he's too unremittingly theoretical), but a casual reader could mistake his intentions.

Second, if you just look at programs, and try to make sense

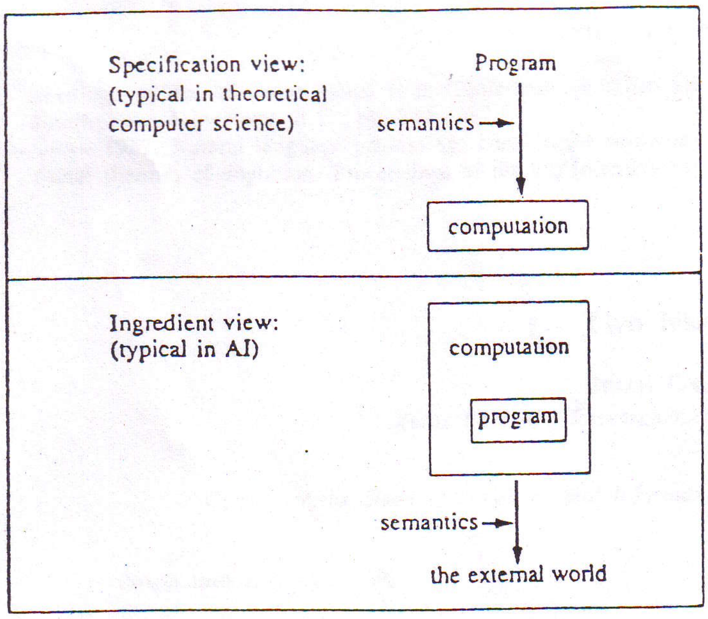

FIG. 2. Two readings of "program".

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 isn't in the machine at all.*

The third problem arises from a curious fact about how practice is currently understood. It's 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. (see, for example, Gordon 1979). Now I firmly believe that all current computational systems — from Amord to Zetalisp — blend both factors we talked about earlier. The only factor of computational systems 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."* 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," suggested in Fig. 2. I have tried to explain these diagrams, and some of their consequences, in Smith (1987); the main insight is that programs, as understood in theoretical computer science, are typically viewed as specifications of computational behaviour, whereas AI (McDermott is clearly of this view) takes them to be constituents within it. On the first (computer science) view, the content of a program is the computation specified; hence, the computation itself is what denotational semantics takes its subject matter to be.

Note, in passing, that all sorts of other otherwise odd things are explained by this analysis. To begin with, it explains why programming language semanticists use term models, initial algebras, and all the rest: they individuate their semantical models (remember the third assumption!) so finely because they are really modelling (the behaviour of) computational processes themselves. In addition, since you are supposed to be able to instantiate a computation, given a program specifying it (programs are "prescriptions," as well as "descriptions," to use Nygaard's phrase), the semantical relation is constrained to be effective in a way that the content relation, for natural language and AI, is clearly not (remember the first lesson!). Furthermore, operational and denotational semantics, in theoretical computer science, are two different kinds of analysis of the same relation; that's why you see equivalence proofs between them (the distinction between operational and denotational semantics, in other words, is completely different from the one we've been talking about between first and second factors). Finally, note that, under this view, a computer really does interpret a program, in the philosopher's sense!

But enough about someone else's worries. The important point here is that the content relation that AI needs to study, as opposed to the case just considered, is the one that holds between the computational process and the world outside it. On the "ingredient" view of programs, this would just be the semantics of the program itself; on the specification (computer science) view, it would be the *semantics of the semantics of the program* (i.e., two levels of reference). Whichever way you go, three things should be remembered about it: it will (at least in general) reach outside the machine, it won't (again, in general) be effective, and it won't ever be computed.

Let's get back 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 three worries. First, if (as a casual reader of McDermott) you just endorse practice, you are liable to remain a-theoretical. 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, doesn't appear in the program at all). And then third, the twister: if you borrow techniques from theoretical computer science, you will be led to focus on the wrong relation completely. Furthermore, not only doesn't it focus on the semantical relation we're 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'll miss out on logic's two great lessons. And that — I hope McDermott will agree — would be tragic.

GORDON, M. 1979. The denotational description of programming languages: an introduction. Springer-Verlag, New York, NY.

ISRAEL, D. 1980. What's wrong with non-monotonic logic? Proceedings of the First Annual National Conference on Artificial Intelligence, Stanford, CA, pp. 99–101.

KRIPKE, S. 1963. Semantical considerations on modal logic. Acta Philosophica Fennica, 16: 83–94.

SMITH, B. C. 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.

——— 1986. Varieties of self-reference. *In* Theoretical aspects of reasoning about knowledge. *Edited by* Joseph Halpern, Morgan Kaufman, Los Altos, CA, pp. 19–43. Also to appear in Artificial Intelligence.

——— 1987. The correspondence continuum. CSLI Technical Report CSLI-87-71, Stanford University, Stanford, CA. Also appeared with the Proceedings of the Sixth Canadian Conference on Artificial Intelligence. Montréal, Quebec, March 1986, and to appear in Artificial Intelligence.