1a Preliminaries

1 Abstract†

We show how a computational system can be constructed to "reason," effectively and consequentially, about its own inferential processes. The analysis proceeds in two parts. First, we consider the general question of computational semantics, rejecting traditional approaches, and arguing that the *declarative* and *procedural* aspects of computational symbols (what they stand for, and what behavior they engender) should be analyzed *independently*, in order that they may be coherently related. Second, we investigate *self-referential* behavior in computational processes, and show how to embed an effective procedural model of a computational calculus within that calculus (a model not unlike a meta-circular interpreter, but connected to the fundamental operations of the machine in such a way as to provide, at any point in a computation, fully articulated descriptions of the state of that computation, for inspection and possible modification). In terms of the theories that result from these investigations, we present a general architecture for **procedurally reflective** processes, able to shift smoothly between dealing with a given subject domain, and dealing with their own reasoning processes over that domain.

An instance of the general solution is worked out in the context of an applicative language. Specifically, we present three successive dialects of Lisp: **1Lisp**,‡ a distillation of current practice, for comparison purposes; **2Lisp**, a dialect constructed in terms of our rationalized semantics, in which the

A1

†The section numbers used here ('1' for the Abstract; '2', Extended Abstract; '3', Preface; and '4', Prologue) were introduced for this version.

‡As indicated in the Cover, I have removed the hyphens from the terms '1-Lisp', '2-Lisp', and '3-Lisp' used in the dissertation, here (and throughout) naming them '1Lisp', '2Lisp', and '3Lisp,' respectively.

concept of evaluation is rejected in favour of independent notions of *simplification* and *reference,* and in which the respective categories of notation, structure, semantics, and behavior arc strictly aligned; and **3Lisp**, an extension of 2Lisp endowed with reflective powers.

2 Extended Abstract

We show how a computational system can be constructed to A2
"reason" effectively and consequentially about its own inference processes. Our approach is to analyze *self-referential* behavior in computational systems, and to propose a theory of *procedural reflection* that enables any programming language to be extended in such a way as to support programs able to access and manipulate structural descriptions of their own operations and structures. In particular, one must encode an explicit theory of such a system within the structures of the system, and then connect that theory to the fundamental operations of the system in such a way as to support three primitive behaviors. First, at any point in the course of a computation, fully articulated descriptions of the state of the reasoning process must be available for inspection and modification. Second, it must be possible at any point to resume an arbitrary computation in accord with such (possibly modified) theory-relative descriptions. Third, procedures that reason with descriptions of the processor state must themselves be subject to description and review, to arbitrary depth. Such *reflective* abilities allow a process to shift smoothly between dealing with a given subject domain, and dealing with its own reasoning processes over that domain.

Crucial in the development of this theory is a comparison of the respective semantics of programming languages (such as Lisp and Algol) and declarative languages (such as logic and the λ-calculus); we argue that unifying these traditionally separate disciplines clarifies both, and suggests a simple

and natural approach to the question of procedural reflection. More specifically, the semantical analysis of computational systems should comprise independent formulations of **declarative import** (what symbols stand for) and **procedural consequence** (what effects and results are engendered by processing them), although the two semantical treatments may, because of side-effect interactions, have to be formulated in conjunction. When this approach is applied to a functional language it is shown that the traditional notion of *evaluation* is confusing and confused, and must be rejected in favour of independent notions of *reference* and *simplification*. In addition, we defend a standard of **category alignment**: there should be a systematic correspondence between the respective categories of notation, abstract structure, declarative semantics, and procedural consequence (a mandate satisfied by no extant procedural formalism). It is shown how a clarification of these prior semantical and aesthetic issues enables a *procedurally reflective* dialect to be clearly defined and readily constructed.

An instance of the general solution is worked out in the context of an applicative language, where the question reduces to one of defining an interpreted calculus able to inspect and A3 affect its own interpretation. In particular, we consider three successive dialects of Lisp: 1Lisp, a distillation of current practice for comparison purposes; 2Lisp, a dialect *categorically* and *semantically rationalized* with respect to an explicit theory of declarative semantics for s-expressions; and 3Lisp, a derivative of 2Lisp endowed with full reflective powers. 1Lisp, like all Lisp dialects in current use, is at heart a *first-order* language, employing meta-syntactic facilities and dynamic variable scoping protocols to partially mimic higher-order functionality. 2Lisp like Scheme and the λ-calculus, is higher-order: it supports arbitrary function designators in argument position, is lexically scoped, and treats the function position of an application in a standard extensional manner. Unlike Scheme, however, the

2Lisp processor is based on a regimen of *normalisation*, taking each expression into a normal-form co-designator of its referent, where the notion of *normal-form* is in part defined with respect to that referent's semantic type, not (as in the case of the λ-calculus) solely in terms of the further non-applicability of a set of syntactic reduction rules. 2Lisp normal-form designators are environment-independent and side-effect free; thus the concept of a *closure* can be reconstructed as a *normal-form function designator*. In addition, since normalisation is a form of simplification, and is therefore *designation-preserving*, meta-structural expressions are not de-referenced upon normalisation, as they are when evaluated. Thus we say that the 2Lisp processor is **semantically flat**, since it stays at a semantically fixed level (although explicit referencing and de-referencing primitives are also provided, to facilitate explicit level shifts). Finally, because of its category alignment, *argument objectification* (the ability to apply functions to a sequence of arguments designated collectively by a single term) can be treated in the 2Lisp base-level language, without requiring resort to meta-structural machinery.

A4 A5

3Lisp is straightforwardly defined as an extension of 2Lisp, with respect to an explicitly articulated procedural theory of 3Lisp embedded in 3Lisp structures. This embedded theory, called the **reflective model**, though superficially resembling a meta-circular interpreter, is causally connected to the workings of the underlying calculus in crucial and primitive ways. Specifically, *reflective procedures* are supported that bind as arguments (designators of) the continuation and environment structure of the processor that would have been in effect at the moment the reflective procedure was called, had the machine been running all along in virtue of the explicit processing of that reflective model. Because reflection may recurse arbitrarily, 3Lisp is most simply defined as an infinite tower of 3Lisp processes, each engendering the process immediately be-

low it. Under such an account, the use of reflective procedures amounts to running programs at arbitrary levels in this reflective hierarchy. Both a straightforward implementation and a conceptual analysis are provided to demonstrate that such a machine is nevertheless finite.

The 3Lisp reflective model unifies three programming language concepts that have formerly been viewed as independent: meta-circular interpreters, explicit names for the primitive interpretive procedures (EVAL and APPLY in standard Lisp dialects), and procedures that access the state of the implementation (typically provided, as part of a programming environment, for debugging purposes). We show how all such behaviors can be defined within a pure version of 3Lisp (i.e., independent of implementation), since all aspects of the state of any 3Lisp process are available, with sufficient reflection, as objectified entities within the 3Lisp **structural field**.

3 Preface

The possibility of constructing a reflective calculus first struck me in June 1976, at the Xerox Palo Alto Research Center (PARC), where I was spending a summer working with the KRL representation language of Bobrow and Winograd.[1] As an exercise to learn the new language, I had embarked on the project of representing KRL in KRL; it seemed to me that this "double-barreled" approach, in which I would have both to *use* and to *mention* the language, would be a particularly efficient way to unravel its intricacies. Though that exercise was ultimately abandoned, I stayed with it long enough to become intrigued by the thought that one might build a system that was self-descriptive in an important way (certainly in a way in which my KRL project was *not*). More specifically, I could dimly envisage a computational system in which what happened took effect in virtue of declarative descriptions of what was to

1. KRL' for 'Knowledge Representation Language; see Bobrow and Winograd (1977) and Bobrow et al. (1977).

happen, and in which the internal structural conditions were represented in declarative descriptions of those internal structural conditions. In such a system a program could with equal ease access all the basic operations and structures either directly or in terms of completely (and automatically) articulated descriptions of them. The idea seemed to me rather simple (as it still does); furthermore, for a variety of reasons I thought that such a reflective calculus could *itself* be rather simple—in some important ways simpler than a non-reflective formalism (this too I still believe). *Designing* such a formalism, however, no longer seems as straightforward as I thought at the time; this dissertation should be viewed as the first report emerging from the research project that ensued.

Most of the five years since 1976 have been devoted to initial versions of my specification of such a language, called **Mantiq**, based on these original hunches. As mentioned in A6
the first paragraph of [dissertation†] chapter 1,‡ there are various non-trivial goals that must be met by the designer of any such formalism, including at least a tentative solution to the knowledge representation problem. Furthermore, in the course of its development, Mantiq has come to rest on some additional hypotheses above and beyond those mentioned above (including, for example, a sense that it will be possible within a computational setting to construct a formalism in which syntactic identity and intensional identity can be identified, given some appropriate, but *independently* specified, A7
theory of intensionality). Probably the major portion of my attention to date has focused on these intensional aspects of the Mantiq architecture.

It was clear from the outset that no dialect of Lisp (or of any other purely procedural calculus) could serve as a full re-

†To minimise confusion I explicitly flag chapter references that refer to chapters in the dissertation, of which only chapter 1 is included in this Volume, so as to distinguish them from references to chapters in the present volume.

‡Included here as ch. 3b, p.·····.

flective formalism; purely declarative languages like logic or the λ-calculus were dismissed for similar reasons. In February of 1981, however, I decided that it would be worth focusing on Lisp, by way of an example, in order to work out the details of a specific subset of the issues with which Mantiq would have to contend. In particular, I recognized that many of the questions of reflection could be profitably studied in a (limited) procedural dialect, in ways that would ultimately illuminate the larger programme. Furthermore, to the extent that Lisp could serve as a theoretical vehicle, it seemed a good project; it would be much easier to develop, and even more so to communicate, solutions in a formalism at least partially understood. A8 A9

The time from the original decision to look at procedural reflection (and its concomitant emphasis on semantics—I realized from investigations of Mantiq that semantics would come to the fore in all aspects of the overall enterprise), to a working implementation of 3Lisp, was only a few weeks. Articulating why 3Lisp was the way it was, however—i.e., formulating in plain English the concepts and categories on which the design was founded—required quite intensive work for the remainder of the year. A first draft of the dissertation was completed at the end of December 1981; the implementation remained essentially unchanged during the course of this writing (the only substantive alteration was the idea of treating recursion in terms of explicit Y-operators). Thus—and I suspect there is nothing unusual in this experience—formulating an idea required approximately ten times more work than embodying it in a machine; perhaps more surprisingly, all of that effort in formulation occurred *after* the implementation was complete. We sometimes hear that writing computer programs is intellectually hygienic because it requires that we make our ideas completely explicit. I have come to disagree rather fundamentally with this view. Certainly writing a pro- A10 A11

gram does not force one to one make one's ideas *articulate*, although it is a useful first step. More seriously, however, it is often the case that the organising principles and fundamental insights contributing to the coherence of a program are not explicitly encoded within the structures comprising that program. The theory of declarative semantics embodied in 3Lisp, for example, was initially tacit—a fact perhaps to be expected, since only procedural consequence is explicitly encoded in an implementation. Curiously, this is one of the reasons that building a fully reflective formalism (as opposed to the limited procedurally reflective languages considered here) is difficult: in order to build a general reflective calculus, one must embed within it a fully articulated theory of one's understanding of it. This will take some time.

4 Prologue

It is a striking fact about human cognition that we can think not only about the world around us, but also about our ideas, our actions, our feelings, our past experience. This ability to **reflect** lies behind much of the subtlety and flexibility with which we deal with the world; it is an essential part of mastering new skills, of reacting to unexpected circumstances, of short-range and long-range planning, of recovering from mistakes, of extrapolating from past experience, and so on and so forth. Reflective thinking characterizes mundane practical matters and delicate theoretical distinctions. We have all paused to review past circumstances, such as conversations with guests or strangers, to consider the appropriateness of our behavior. We can remember times when we stopped and consciously decided to consider a set of options, say when confronted with a fire or other emergency. We understand when someone tells us to believe everything a friend tells us, unless we know otherwise. In the course of philosophical discussion we can agree to distinguish views we believe to be true

from those we have no reason to believe are false. In all these cases the subject matter of our contemplation at the moment of reflection includes our remembered experience, our private thoughts, and our reasoning patterns.

The power and universality of reflective thinking has caught the attention of the cognitive science community—indeed, once alerted to this aspect of human behavior, theorists find evidence of it almost everywhere. Though no one can yet say just what it comes to, crucial ingredients would seem to be the ability to recall memories of a world experienced in the past and of one's own participation in that world, the ability to think about a phenomenal world, hypothetical or actual, that is not currently being experienced (an ability presumably mediated by our knowledge and belief), and a certain kind of true self-reference: the ability to consider both one's actions and the workings of one's own mind. This last aspect—the self-referential aspect of reflective thought—has sparked particular interest for cognitive theorists, both in psychology (under the label *meta-cognition)* and in artificial intelligence (in the design of computational systems possessing inchoate reflective powers, particularly as evidenced in a collection of ideas loosely allied in their use of the term "meta": meta-level rules, meta-descriptions, and so forth).

In artificial intelligence, the focus on computational forms of self-referential reflective reasoning has become particularly central. Although the task of endowing computational systems with subtlety and flexibility has proved difficult, we have had some success in developing systems with a moderate grasp of certain domains: electronics, bacteremia, simple mechanical systems, etc. One of the most recalcitrant problems, however, has been that of developing flexibility and modularity (in some cases even simple effectiveness) in the reasoning processes that use this world knowledge. Though it has been possible to construct programs that perform a specific kind of

reasoning task (say, checking a circuit or parsing a subset of natural language syntax), there has been less success in simulating "common sense," or in developing programs able to figure out what to do, and how to do it, in either general or novel situations. If the course of reasoning—if the problem solving strategies and the hypothesis formation behavior—could *itself* be treated as a valid subject domain in its own right, then (at least so the idea goes) it might be possible to construct systems that manifested the same modularity about their own thought processes that they manifest about their primary subject domains. A simple example might be an electronics "expert" able to choose an appropriate method of tackling a particular circuit, depending on a variety of questions about the relationship between its own capacities and the problem at hand: whether the task was primarily one of design or analysis or repair, what strategies and skills it knew it had in such areas, how confident it was in the relevance of specific approaches based on, say, the complexity of the circuit, or on how similar it looked compared with circuits it already knew. Expert human problem-solvers clearly demonstrate such reflective abilities, and it appears more and more certain that powerful computational problem solvers will have to possess them as well.

No one would expect potent skills to arise automatically in a reflective system; the mere *ability* to reason about the reasoning process will not magically yield systems able to reflect in powerful and flexible ways. On the other hand, the demonstration of such an ability is clearly a prerequisite to its effective utlization. Furthermore, many reasons are advanced in support of reflection, as well as the primary one (the hope of building a system able to decide how to structure the pattern of its own reasoning). It has been argued, for example, that it would be easier to construct powerful systems in the first place (it would seem you could almost *tell them* how to think), to interact with them when they fail, to trust them if

they could report on how they arrive at their decisions, to give them "advice" about how to improve or discriminate, as well as to provide them with their own strategies for reacting to their history and experience.

There is even, as part of the general excitement, a tentative suggestion on how such a self-referential reflective process might be constructed. This suggestion—nowhere argued but clearly in evidence in several recent proposals—is a particular instance of a general hypothesis, adopted by most A.I. researchers, that we will call the *Knowledge Representation Hypothesis*. It is widely held in computational circles that any process capable of reasoning intelligently about the world must consist in part of a field of structures, of a roughly linguistic sort, which in some fashion *represent* whatever knowledge and beliefs the process may be said to possess. For example, according to this view, since I know that the sun sets each evening, my "mind" must contain (among other things) a language-like or symbolic structure that represents this fact, inscribed in some kind of internal code. There are various assumptions that go along with this view: there is for one thing presumed to be an internal process that "runs over" or "computes with" these representational structures, in such a way that the intelligent behavior of the whole results from the interaction of parts. In addition, this ingredient process is required to react only to the "form" or "shape" of these mental representations, without regard to what they mean or represent—this is the substance A12 of the claim that computation involves *formal* symbol manipulation. Thus my thought that, for example, the sun will soon set, would be taken to emerge from an interaction in my mind between an ingredient process and the shape or "spelling" of various internal structures representing my knowledge that the sun does regularly set each evening, that it is currently tea time, and so forth.

The knowledge representation hypothesis may be summa-

rized as follows:

> **Knowledge Representation Hypothesis:** Any mechanically embodied intelligent process will be comprised of structural ingredients that (a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and (b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge. A12.5

Thus for example if we felt disposed to say that some process knew that dinosaurs were warm-blooded, then we would find (according, presumably, to the best explanation of how that process worked) that a certain computational ingredient in that process was understood as *representing* the (propositional) fact that dinosaurs were warm-blooded, and furthermore, that this very ingredient played a role, independent of our understanding of it as representational, in leading the process to behave in whatever way inspired us to say that it knew that fact. Presumably we would be convinced by the manner in which the process answered certain questions about their likely habitat, by assumptions it made about other aspects of their existence, by postures it adopted on suggestions as to why they may have become extinct, etc.

A careful analysis will show that. to the extent that we can make sense of it, this view that *knowing is representational* is far less evident—and perhaps, therefore, far more interesting—than is commonly believed. To do it justice requires considerable care: accounts in cognitive psychology and the philosophy of mind tend to founder on simplistic models of computation, and artificial intelligence treatments often lack the theoretical rigour necessary to bring the essence of the idea into plain view. Nonetheless, conclusion or hypothesis, it permeates cur-

rent theories of mind, and has in particular led researchers in artificial intelligence to propose a spate of computational languages and calculi designed to underwrite such representation. The common goal is of course not so much to speculate on what is actually represented in any particular situation as to uncover the general and categorical form of such representation. Thus no one would suggest how anyone actually represents facts about tea and sunsets: rather, they might posit the general form in which such beliefs would be "written" (along with other beliefs, such as that Lhasa is in Tibet, and that π is an irrational number). Constraining all plausible suggestions, however, is the requirement that they must be able to demonstrate how a particular thought could emerge from such representations—this is a crucial meta-theoretic characteristic of artificial intelligence research. It is traditionally considered insufficient merely to propose true theories that do not enable some causally effective mechanical embodiment. The standard against which such theories must ultimately judged, in other words, is whether they will serve to underwrite the construction of demonstrable, behaving artefacts. Under this general rubric knowledge representation efforts differ markedly in scope, in approach, and in detail; they differ on such crucial questions as whether or not the mental structure are modality specific (one for visual memory, another for verbal, for example). In spite of such differences, however, they manifest the shared hope that an attainable first step towards a full theory of mind will be the discovery of something like the structure of the "mechanical mentalese" in which our beliefs are inscribed.

It is natural to ask whether the knowledge representation hypothesis deserves our endorsement, but this is not the place to pursue that difficult question. Before it can fairly be asked, we would have to distinguish a strong version claiming that knowing is *necessarily* representational from a weaker version

claiming merely that it is *possible* to build a representational knower. We would run straight into all the much-discussed but virtually intractable questions about what would be required to convince us that an artificially constructed process exhibited intelligent behavior. We would certainly need a definition of the word 'represent,' about which we will subsequently have a good deal to say. Given the current (minimal) state of our understanding, I myself see no reason to subscribe to the A13 strong view, and remain skeptical of the weak version as well. But one of the most difficult questions is merely to ascertain what the hypothesis is actually saying—thus my interest in representation is more a concern to *make it clear* than it is to defend or deny it The entire present investigation, therefore, will be pursued under this hypothesis, not because we grant it our allegiance, but merely because it deserves our attention.

Given the representation hypothesis, the suggestion as to how to build self-reflective systems—a suggestion we will call the *Reflection Hypothesis*—can be summarized as follows:

> **Reflection Hypothesis:** In as much as a computational process can be constructed to reason about an external world in virtue of comprising an ingredient process (interpreter) formally manipulating representations of that world, so too a computational process could be made to reason about itself in virtue of comprising an ingredient process (interpreter) formally manipulating representations of its own operations and structures.

Thus the task of building a computationally reflective system is thought to reduce to, or at any rate to include, the task of providing a system with formal representations of its own constitution and behavior. Hence a system able to imagine a world where unicorns have wings would have to construct formal representations of that fact; a system considering the

adoption of a hypothesis-and-test style of investigation would have to construct formal structures representing such an inference regime.

Whatever its merit, there is ample evidence that researchers arc taken with this view. Systems such as Weyhrauch's FOL, Doyle's TMS, McCarthy's ADVICE-TAKER, Hayes' GOLUM, and Davis' TERESIUS arc particularly explicit exemplars of just such an approach.[2] In Weyhrauch's system, for example, sentences in first-order logic arc constructed that axiomatize the behavior of the Lisp procedures used in the course of the computation (FOL is a prime example of the dual-calculus approach mentioned earlier). In Doyle's systems, explicit representations of the dependencies between beliefs and of the "reasons" the system accepts a conclusion play a causal role in the inferential process. Similar remarks hold for the other projects mentioned, as well as for a variety of other current research. In addition, it turns out on scrutiny that a great deal of current computational practice can be seen as dealing, in one way or another, with reflective abilities, particularly as exemplified by computational structures representing other computational structures. We constantly encounter examples: the wide-spread use of macros in Lisp, the use of meta-level structures in representation languages, the use of explicit non-monotonic inference rules, the popularity of meta-level rules in planning systems.[3] Such a list can be extended indefinitely; in a recent symposium Brachman reported that the love affair with *"meta-level reasoning"* was the most important theme of knowledge representation research in the last decade.[4]

A14

2. Weyhrauch (1978), Doyle (1979), McCarthy (1968), Hayes (1979), and Davis (1980a), respectively.

3. For a discussion of macros see the various sources on Lisp mentioned in note 16 of chapter 1; meta-level rules in representation were discussed in Brachman and Smith (1980); for a collection of papers on non-monotonic reasoning see Bobrow (1980); macros are discussed in Pitman (1980).

4. Brachman (1980).

4a The Relationship Between Reflection & Representation

The manner in which this discussion has been presented so far would seem to imply that the interest in *reflection* and the adoption of a *representational* stance are theoretically independent positions. I have argued in this way for a reason: to make clear that the two subjects are not the same. There is no *a priori* reason to believe that even a fully representational system should in any way be reflective or able to make anything approximating a reference to itself; similarly, there is no *proof* that a powerfully self-referential system need be constructed of representations. However—and this is the crux of the matter—the reason to raise both issues together is that they are surely, in some sense, related. If nothing else, the word 'representation' comes from 're' plus 'present', and the ability to *re-present* a world to itself is undeniably a crucial, if not *the* crucial, ingredient in reflective thought. If I reflect on my childhood, I re-present to myself my school and the rooms of my house; if I reflect on what I will do tomorrow, I bring into the view of my mind's eye the self I imagine that tomorrow I will be. If we take "representation" to describe an *ability* rather than a *structure*, reflection surely involves representation (although—and this should be kept clearly in mind—the "representation" of the knowledge representation hypothesis refers to ingredient structures, not to an activity). A15

It is helpful to look at the historical association between these ideas, as well to search for commonalities in content. In the early days of artificial intelligence, a search for the general patterns of intelligent reasoning led to the development of such general systems as Newell and Simon's GPS, predicate logic theorem provers, and so forth.[5] The descriptions of the subject domains were minimal but were nonetheless primarily declarative, particularly in the case of the systems based on logic. However it proved difficult to make such gen-

5. Newell and Simon (1963); Newell and Simon (1956).

eral systems effective in particular cases: so much of the "expertise" involved in problem solving seems domain and task specific. In reaction against such generality, therefore, a *procedural* approach emerged in which the primary focus was on the manipulation and reasoning about specific problems in simple worlds.[6] Though the procedural approach in many ways solved the problem of undirected inferential meandering, it too had problems: it proved difficult to endow systems with much generality or modularity when they were simply constituted of procedures designed to manifest certain particular skills. In reaction to such brittle and parochial behavior, researchers turned instead to the development of processes designed to work over general representations of the objects and categories of the world in which the process was designed to be embedded. Thus the *representation hypothesis* emerged in the attempt to endow systems with generality, modularity, flexibility, and so forth with respect to the embedding world, but to retain a procedural effectiveness in the control component.[7] In other words, in terms of our main discussion, representation as a method emerged as a solution to the problem of providing general and flexible ways of reflecting (not self-referentially) about the world. A16

Systems based on the representational approach—and it is fair to say that most of the current "expert systems" are in this tradition—have been relatively successful in certain respects, but a major lingering problem has been a narrowness and inflexibility regarding the style of reasoning these systems employ in using these representational structures. This inflexibility in *reasoning* is strikingly parallel to the inflexibility in *knowledge* that led to the first round of representational systems; researchers have therefore suggested that we need re-

6. The proceduralist view was represented particularly by a spate of dissertations emerging from MIT at the beginning of the 1970s; see for example Winograd (1972), Hewitt (1972), Sussman et al. (1971), etc.
7. See Minsky (1975), Winograd (1975), and all of the systems reported in Brachman and Smith (1980).

flective systems able to deal with their own constitutions as well as with the worlds they inhabit. In other words, since the *style* of the problem is so parallel to that just sketched, it has seemed that another application of the same medicine might be appropriate. If we could inscribe general knowledge about how to reason in a variety of circumstances in the "mentalese" of these systems, it might be possible to design a relatively simpler inferential regime over this "meta-knowledge about reasoning," thereby engendering a flexibility and modularity regarding reasoning, just as the first representational work engendered a flexibility and modularity about the process's embedding world.

There are problems, however, in too quick an association between the two ideas, not the least of which is the question of to *whom* these various forms of re-presentation are being directed. In the normal case—that is to say, in the typical computational process built under the aegis of the knowledge representation hypothesis—a process is constituted from symbols that we as external theorists take to be representational structures; they are visible *only to the ingredient interpretive process [that is just part] of the whole*, and they are visible to that constituent process *only formally* (this is the basic claim of computation). Thus the interpreter can see them, though it is blind to the fact of their being representations. (In fact it is almost a great joke that the blindly formal ingredient process A17 should be called an *interpreter*: when the Lisp interpreter evaluates the expression '(+ 2 3)' and returns the result '6', the last thing it knows is that the numeral '2' denotes the number *two*.)

Whatever is the case with the ingredient process, there is no reason to suppose that the representational structures are visible to the whole constituted process *at all*, formally or informally. That process is made out of them; there is no more *a priori* reason to suppose that they are accessible to its in- A18

spection than to suppose that a camera could take a picture of its own shutter—no more reason to suppose it is even a coherent possibility than to say that France is near Marseilles. A19 Current practice should overwhelmingly convince us of this point: what is as tacit—what is as thoroughly lacking in self-knowledge—as the typical modern computer system?

The point of the argument here is not to prove that one *cannot* make such structures accessible—that one *cannot* make a representational reflective system—but to make clear that two ideas are involved. Furthermore, they are different in kind: one (representation) is a possibly powerful *method* for the construction of systems; the other (reflection) is a kind of *behavior* we are asking our systems to exhibit. It remains a question whether the representational method will prove useful in the pursuit of the goal of reflective behavior.

That, in a nutshell, is our overall project.

4b The Theoretical Backdrop

It takes only a moment's consideration of such questions as the relationship between representation and reflection to recognize that the current state of our understanding of such subjects is terribly inadequate. In spite of the general excitement about reflection, self-reference, and computational representation, no one has presented an underlying theory of any of these issues. The reason is simple: we are so lacking in adequate theories of the surrounding territory that, without considerable preliminary work, cogent definitions cannot even be attempted. Consider for example the case regarding self-referential reflection, where just a few examples will make this clear.

1. From the fact that a reflective system A is implemented in system B, it docs not follow that system B is thereby rendered reflective (for example, in this dissertation I will present a partially-reflective dialect of Lisp that I

have implemented on a Digital Systems Corporation PDP-10, but the PDP-10 is not itself reflective). Hence even a *definition* of reflection will have to be backed by theoretical apparatus capable of distinguishing between one abstract machine and another in which the first is implemented—something we are not yet able to do. A20

2. The notion seems to require of a computational process, and (if we subscribe to the representational hypothesis) of its interpreter, that in reflecting it "back off" one level of reference, and we lack theories both of interpreters in general, and of computational reference in particular.
3. Theories of computational interpretation will be required to clarify the confusion mentioned above regarding the relationship between reflection and representation: for a system to reflect it must re-present *for itself* its mental states; it is not sufficient for it to comprise a set of formal representations inspected *by its interpreter.* This is a distinction we encounter again and again; a failure to make it is the most common error in discussions of the plausibility of artificial intelligence from those outside the computational community, derailing the arguments of such thinkers as Searle and Fodor.[8] A21
4. Theories of reference will be required in order to make sense of the question of what a computational process is "thinking" about at all, whether reflective or not (for example, it may be easy to claim that when a program is manipulating data structures representing women's votes that the process as a whole is "thinking about suffrage," but what is the process thinking about when the interpreter is expanding a macro definition?).

8. Searle (1980), Fodor (1978 and 1980).

5. Finally, if the search for reflection is taken up too enthusiastically, one is in danger of interpreting everything as evidence of reflective thinking, since what may not be reflective *explicitly* can usually be treated as *implicitly* reflective (especially given a little imagination on the part of the theorist). However we lack general guidelines on how to distinguish explicit from implicit aspects of computational structures. A22

Nor is our grasp of the representational question any clearer; a serious difficulty, especially since the representational endeavour has received much more attention than has reflection. Evidence of this lack can be seen in the fact that, in spite of an approximate consensus regarding the general form of the task, and substantial effort on its behalf, no representation scheme yet proposed has won substantial acceptance in the field. A23 Again this is due at least in part to the simple absence of adequate theoretical foundations in terms of which to formulate either enterprise or solution. We do not have theories of either representation or computation in terms of which to define the terms of art currently employed in their pursuit (*representation, implementation, interpretation, control structure. data structure, inheritance,* and so forth), and are consequently without any well-specified account of what it would be to succeed, let alone of what to investigate, or of how to proceed. Numerous related theories have been developed (model theories for logic, theories of semantics for programming languages, and so forth), but they do not address the issues of knowledge representation directly, and it is surprisingly difficult to weave A24 their various insights into a single coherent whole.

The representational consensus alluded to above, in other words, is widespread but vague; disagreements emerge on every conceivable technical point, as was demonstrated in a recent survey of the field.[9] To begin with, the central notion of "representation" remains notoriously unspecified: in spite

9. Brachman and Smith (1980).

of the intuitions mentioned above, there is remarkably little agreement on whether a representation must "re-present" in any constrained way (like an image or copy), or whether the word is synonymous with such general terms as "sign" or "symbol." A further confusion is shown by an inconsistency in usage as to what representation is a relationship between. The sub-discipline is known as the *representation of knowledge*, but in the survey just mentioned by far the majority of the respondents (to the surprise of this author) claimed to use the word, albeit in a wide variety of ways, as between formal symbols *and the world about which the process is designed to reason*. Thus a KLONE structure might be said to *represent Don Quixote tilting at a windmill*; it would not be taken as representing *the fact or proposition of this activity*. In other words the majority opinion is not that we are *representing knowledge* at all, but rather, as we put it above, that *knowing is representational*.[10] A25

In addition, we have only a dim understanding of the relationship that holds between the purported representational structures and the ingredient process that interprets them. This relates to the crucial distinction between that interpreting process and the whole process of which it is an ingredient (whereas it is *I* who thinks of sunsets, it is at best a *constituent of my mind* that inspects a mental representation). Furthermore, there are terminological confusions: the word 'semantics' is applied to a variety of concerns, ranging from how natural language is translated into the representational structures, to what those structures represent, to how they impinge on the rational policies of the "mind" of which they are a part, to what functions are computed by the interpreting process, etc. The term 'interpretation' (to take another example) has two relatively well-specified but quite independent meanings, one of computational origin, the other more philosophical; how the two relate remains so far unexplicated, although, as was just mentioned, they are strikingly distinct. A26 A27

10. See the introduction to Brachman and Smith (1980).

Unfortunately, such general terminological problems are just the tip of an iceberg. When we consider our specific representational proposals, we are faced with a plethora of apparently incomparable technical words and phrases. *Node, frame, unit, concept, schema, script, pattern, class,* and *plan,* for example, are all popular terms with similar connotations and ill-defined meaning.[11] The theoretical situation (this may not be so harmful in terms of more practical goals) is further hindered by the tendency for representational research to be reported in a rather demonstrative fashion: researchers typically exhibit particular formal systems that (often quite impressively) embody their insights, but that are defined using formal terms peculiar to the system at hand. We are left on our own to induce the relevant generalities and to locate them in our evolving conception of the representation enterprise as a whole. Furthermore, such practice makes comparison and discussion of technical details always problematic and often impossible, defeating attempts to build on previous work.

This lack of grounding and focus has not passed unnoticed: in various quarters one hears the suggestion that, unless severely constrained, the entire representation enterprise may be ill-conceived—that we should turn instead to considerations of particular epistemological issues (such as how we reason about, say, liquids or actions), and should use as our technical base the traditional formal systems (logic, Lisp, and so forth) that representation schemes were originally designed to replace.[12] In defense of this view two kinds of argument are often advanced. The first is that questions about the *central* cognitive faculty are at the very least premature, and more seriously may for principled reasons never succumb to the kind of rigorous scientific analysis that characterizes recent studies of the *peripheral* aspects of mind: vision, audition, gram-

11. References on *node, frame, unit, concept, schema, script, pattern, class,* and *plan* can be found in the various references provided in Brachman and Smith (1980).

12. See in particular Hayes (1978).

mar, manipulation, and so forth.[13] The other argument is that logic as developed by the logicians is in itself sufficient; that all we need is a set of ideas about what axioms and inference protocols are best to adopt.[14] But such doubts cannot be said to have deterred the whole of the community: the survey just mentioned lists more than thirty new representation systems A28 under active development.

The strength of this persistence is worth noting, especially in connection with the theoretical difficulties just sketched. There can be no doubt that there are scores of difficult problems: we have just barely touched on some of the most striking. But it would be a mistake to conclude in discouragement that the *enterprise* is doomed, or to retreat to the meta-theoretic stability of adjacent fields (like proof theory, model theory, programming language semantics, and so forth). The moral is at once more difficult and yet more hopeful. What is demanded is that we stay true to these undeniably powerful ideas, and attempt to develop adequate theoretical structures on this home ground. It is true that any satisfactory theory of computational reflection must ultimately rest, more or less explicitly, on theories of computation, of intensionality, of objectification, of semantics and reference, of implicitness, of formality, of computation, of interpretation, of representation, and so forth. On the other hand as a community we have a great deal of practice that often embodies intuitions that we are unable to formulate coherently. The wealth of programs and systems we have built often betray—sometimes in surprising ways—patterns and insights that eluded our conscious thoughts in the course of their development. What is mandated is a *rational reconstruction* of those intuitions and of that practice.

In the case of designing reflective systems, such a recon-

13. The distinction between central and peripheral aspects of mind is articulated in Nilsson (1981); on the impossibility of central AI (Nilsson himself feels that the central faculty will quite definitely succumb to AI's techniques) see Dreyfus (1972) and Fodor (1980 and forthcoming).
14. Nilsson (1981).

struction is curiously urgent. In fact this long introductory story ends with an odd twist—one that "ups the ante" in the search for a carefully formulated theory, and suggests that practical progress will be impeded until we take up the theoretical task. In general, it is of course possible (some would even advocate this approach) to build an instance of a class of artefact before formulating a theory of it. The era of sail boats, it has often been pointed out, was already drawing to a close just as the theory of airfoils and lift was being formulated—the theory that, at least at the present time, best explains how those sailboats worked. However there are a number of reasons why such an approach may be ruled out in the present case. For one thing, in constructing a reflective calculus one must support arbitrary levels of meta-knowledge and self-modelling, and it is self-evident that confusion and complexity will multiply unchecked when one adds such facilities to an only partially understood formalism. It is simply likely to be unmanageably complicated to *attempt* to build a A29 self-referential system unaided by the clarifying structure of a prior theory. The complexities surrounding the use of APPLY in Lisp (and the caution with which it has consequently come to be treated) bear witness to this fact. However there is a more serious problem. If one subscribes to the knowledge representation hypothesis, it becomes an integral part of developing self-descriptive systems to provide, encoded within the representational medium, an account of (roughly) the syntax, semantics, and reasoning behavior of that formalism. In other words, if we are to build a process that "knows" about itself: and *if we subscribe to the view that knowing is representational,* then we are committed to providing that system with a *representation* of the self-knowledge with which we aim to endow it. That is, we must have an adequate theories of computational representation and reflection *explicitly formulated,* since *an encoding of that theory is mandated to play a causal role as an actual*

ingredient in the reflective device.

Knowledge of any sort—and self-knowledge is no exception—is always theory relative. The representation hypothesis implies that our theories of reasoning and reflection must be explicit. We have argued that this is a substantial, if widely accepted, hypothesis. One reason to find it plausible comes from viewing the entire enterprise as an attempt to communicate our thought patterns and cognitive styles—including our reflective abilities—to these emergent machines. It may at some point be possible for understanding to be tacitly communicated between humans and system they have constructed. In the meantime, however, while we humans might make do with a rich but unarticulated understanding of computation, representation, and reflection, we must not forget that computers do not [yet] share with us our tacit understanding of what they are. A30