Massachusetts Institute of Technology
Artificial Intelligence Laboratory
Cambridge, Massachusetts, 02139, U.S.A.

Working Paper 174 November 1978

How Is a Knowledge Representation System Like a Piano?

A Questionnaire on Knowledge Representation Research

Brian Cantwell Smith

In the summer of 1978 a decision was made to devote a special issue of the SIGART newsletter to the subject of knowledge representation research. To assist in ascertaining the current state of people's thinking on this topic, the editors (Ron Brachman and myself) decided to circulate an informal questionnaire among the representation community. What was originally planned as a simple list of questions eventually developed into the current document, and we have decided to issue it as a report on its own merits. The questionnaire is offered here as a potential aid both for understanding knowledge representation research, and for analysing the philosophical foundations on which that research is based.

The questionnaire consists of two parts. Part I focuses first on specific details, but moves gradually towards more abstract and theoretical questions regarding assumptions about what knowledge representation is; about the role played by the computational metaphor; about the relationships among model, theory, and program; etc. In part II, in a more speculative vein, we set forth for consideration nine hypotheses about various open issues in representation research.

The research reported here was supported by National Institutes of Health Grant No. 1 P41 RR 01096-02 from the Division of Research Resources, and was conducted at the Artificial Intelligence Laboratory of the Massachusetts Instititute of Technology.

How Is a Knowledge Representation System Like a Piano?

A Questionnaire on Knowledge Representation Research

When we first discussed the idea of editing a special issue of the SIGART newsletter on knowledge representation, we felt that there were two useful functions such an issue could serve: first, to present a coherent account of what knowledge representation is; and second, to identify how each research project could be understood in terms of this analysis. Unfortunately, we did not ourselves have a clear understanding of the current state of representation research. Hence a questionnaire.

Two types of question immediately suggested themselves, neither of which seemed sufficient. We first considered questions focusing on details and particulars; we imagined asking people how they handled such issues as quantifier scope and multiple inheritance, whether they used lexical or graphic notation, etc. Clearly some questions of this sort are valuable, and have been included, but we felt that the questionnaire should go further. Since we are not convinced that representation research is carried out within a generally shared framework, we didn't want simply to inquire after endless details.

The second class of question we considered included more comprehensive ones, such as: "What do you think knowledge representation is all about?", and "What are you basically trying to do?". The problem with general questions, however, is that they invite equally general answers. We felt, in fact, that such an approach would elicit responses no more tightly integrated than a collection of individual research papers.

Finally, therefore, we settled on a third tack: we have attempted to include questions which, in terms of content, are fairly high level and general, but which are nonetheless posed in a relatively specific and precise way. These questions arise from genuine curiosity: we don't think they have "correct" answers; nor have we suggested what our own answers would be. (In fact we are unsure of our answers to some, and would be hard-pressed to defend our answers to others.) Nonetheless, the questions betray us: just by articulating them we have inevitably made reference to our own meta-theoretic framework, and have consequently revealed many of our biases.

It is possible, therefore, that you will find some of the questions dissonant, and unrelated to your conception of representation research. If such is the case, instead of responding directly, we would like you to outline how your understanding differs and suggest how you would have proceeded. Such "answers" will be more valuable, in ascertaining the "lay of the land", than half-hearted responses to questions that don't sit comfortably.

We hope that the answers will reveal a coherent sense, among researchers, of the common enterprise, or -- what would be more interesting -- several distinct, but individually coherent, conceptions. Since we are trying to uncover decisions and judgments people have made implicitly, we don't at all expect people to defend their answers, or even to be able to do so. Our goal is simply to try to articulate the positions that people already hold; the more wide ranging the responses, therefore, the better. If we find no coherence, we will of course be able to conclude no more than that we have failed -- no survey could report definitively that there is no structure in a given discipline. But such an outcome seems unlikely.

The questionnaire consists of two parts, of which the first is fairly straightforward, the second more speculative. Part I is in turn divided into four sections: in the first two the questions are introductory and specific, whereas those in the third are more abstract. In the fourth we ask you to score your reactions to a tabulated list of common representation issues. Part II departs from a strict "question-answer" format: instead we put forward nine rather strong hypotheses about representation research in general, and ask you for your reactions.

Finally: The questionnaire is obviously very long -- it may take an hour or two just to read. One of the reasons the questions are formulated at such length, however, is to enable you to answer them briefly. In many cases a few words should be sufficient to indicate your response. Nonetheless, we appreciate the amount of time it will involve, and we would very much like to thank you in advance for giving it what time you can.

Acknowledgements

This questionnaire, and the ideas presented in it, come together from several sources. The original decision to circulate a questionnaire was taken jointly by myself and Ronald Brachman, in the context of editing a special issue of the SIGART newsletter on knowledge representation. With the help of Marilyn Matz, we drafted an initial list of questions during the summer of 1978. Subsequently, I developed the questionnaire into its current state, expanding the original list of questions into what are now sections A and B of part I, and adding other issues arising from my work on knowledge representation semantics. Many of the ideas set forth in section C of part I, and in part II, were originally proposed and debated in an informal discussion group which met at MIT during the spring of 1978; thanks and credit are due the other participants in those sessions: Mitch Marcus, Bob Berwick, Ron Brachman, Beth Levin, Dave McDonald, Bruce Roberts, Andrew Schulert, Candy Sidner, Guy Steele, and Kurt Van Lehn. Finally, David Levy and Barbara Grosz deserve mention for their partnership in the continuing exploration of these ideas, and for their help in the attempt to articulate them here.

In the version of the questionnaire circulated by mail in November of 1978, there was a note indicating that copies of it would be made available as M.I.T. AI Lab memo 497. A decision was subsequently made to release it instead as working paper 174. This however, change, has no bearing on content: this working paper is identical in content to what was to have been released as a memo.

Part I -- The Main Questionnaire

Section A -- Background

1. With what representation system or research group are you associated? Is your involvement with this system primarily from a user's standpoint, or are you involved in its design? With what other systems have you had substantial experience?

2. What is the best documentation on the representation system you work with? What are the best accounts of your own work? What papers, written by you or others, best articulate your basic framework or overall viewpoint?

3. What, very briefly, are your major areas of interest, and intellectual goals, outside the area of knowledge representation? How does your interest in knowledge representation fit into this overall conception? How did you first come to focus your attention on representation issues? Is representation work currently your main concern?

4. Are you prepared to have your responses publically associated with your name, or would you like them to be kept confidential? If you are willing to be cited, can we take your responses as authoritative regarding the system, or representative of the group, mentioned in question 1? (We intend to use responses in a primarily collective way, in order to identify general trends and positions held. In addition however, if it seems instructive, we might like to associate specific views with specific projects and people. Requests for confidentiality, however, will of course be strictly honoured.)

Section B -- Questions On Your Particular System

To simplify wording, the questions in this section assume that all your representation work is embodied in one specific system. If this is not even approximately the case, you may wish to identify several different systems or projects at the outset, and then refer to them explicitly in answering the questions.

Some of the questions here point to questions in the next section, usually to longer discussions of the same or similar issues.

5. What do you take to be the goals of your particular research system?

6. Is your system designed for a specific use? (For example, is it primarily designed to serve as the framework for a medical decision making program? Is it aimed as an exploration into reasoning about causality and action? etc.) Would you characterize it more as a tool for people building research systems, or more as a self-contained investigation of theoretical issues?

7. On what particular questions are you currently working? (As detailed and specific a list as possible would be appreciated, such as:
   a: Inference mechanisms to reason with intensional set descriptions.
   b: Control of reasoning processes using defaults.
   c: Adjudication of conflicting inheritances arising through multiple description.
   d: Common sense reasoning with negation.
   e: Investigating what kinds of question are answerable using parallel marking schemes.

   etc.)

8. To what domains has your representation system been applied? Why?

9. Are there people who use your system outside of your immediate research group? If so, roughly characterize what their interests are, how they are using it, etc.

10. (Cf. question 30.) Have you constructed, or are you in the process of constructing, any representation "packages" or "modules" which you could distribute for general community use?

11. Would you characterize your system more as
    a: a declarative language,
    b: a language plus an interpreter, or
    c: a language and a set of utility subroutines for manipulating structures in it?

12. To what extent is your representation system implemented? In what language?

13. Does your system handle quantification? If so, does the way in which you deal with it differ substantially from the way it is addressed in predicate logic? If so, how? (For example, you might leave the "scope" of quantifiers either ambiguous or unrepresented.)

14. Does your system employ special techniques for dealing with contexts? If so, in what situations do you use them? (Focus of attention? Belief models? Hypotheticals? Contingent descriptions?) What is the basic context mechanism?

15. Various languages (FRL, KRL, KLONE, etc.) require the user to use the implementation language (usually LISP) to encode the routines that manipulate the declarative representation structures. Is this true of your system, or does it include its own language for procedural specification?

16. Does your system use any form of procedural attachment? If so, what is the basic mechanism of attachment? Borrowing KRL terminology, what traps and triggers do you honour; what servants and demons? (E.g. WHEN-INSTANTIATED, TO-IDENTIFY, WHEN-DESCRIBED, etc.) Do you provide standard subroutines to be called when the traps and triggers are activated, or do you have users write their own programs for these cases?

17. In representing facts about a paper, one of the things you might want to say is something like "the thesis of this paper is that dinosaurs were warm-blooded". Imagine a "frame-like" system, having a frame called "PAPER" with a slot called "THESIS". If you were "filling" this

slot in the example just given, you would need to fill it not with a description of dinosaurs, warm-bloodedness, or anything of that sort; rather, you would need to fill it with (the representation of) the proposition "*that* dinosaurs were warm-blooded". How does your system deal with this kind of explicit use of propositions?

18. Various systems organize their deductive components around one or more central ideas; pattern matching, for example, has sometimes served this purpose. The various inferential strategies are then defined in terms of this underlying conception. Is the reasoning part of your system organized around any such axes? If so, what are they?

19. In LISP, quotation is defined in terms of protection against one application of an evaluation process. We suggest in question 38 that, for representation work, we do not have any obvious notion of an evaluation process ready at hand. Quotation must therefore mean something else in representation systems. What do you think quoted descriptions are? Do you deal with them in your system, and if so what do they mean?

20. Suppose we characterize "meta-descriptions" (or "meta-symbols") to be descriptions (or symbols) whose referents are other symbols or structures within the representation system (cf. question 38, and hypothesis 9 in part II). Do you make use of any such meta-descriptions in your work?

21. Have you encoded a representation of your system in itself? of just the syntactic language? If so, does the system make any substantial use of this recursive/ reflexive/meta-circular description? How? If not, do you plan to construct such an encoding, and do you plan to have your system make use of it?

22. In your system, do the representation structures themselves bear any assertional weight (i.e. does the existence of a structure imply anything about whether the proposition it represents is taken to be true)? If not, how do you assert anything? If so, how do you deal with hypotheticals, negation, and quoted descriptions?

23. (Cf. hypothesis 4 in part II.) In most systems based on predicate logic, when the systems are defined the predicate letters and constants are left "uninterpreted", in the sense that the predicates and individuals which they denote are left unspecified. A certain number of signs, however, are given a fixed interpretation: the truth-functional connectives (AND, OR, NOT, IMPLIES, etc.), and the quantifiers, for example. In other words, no matter what the application, these signs always mean the same thing. In your system, what signs, if any, are given a fixed interpretation?

24. Which other systems in the field are most similar to yours? Which are most different? Very briefly, what do you see to be the major similarities and differences?

25. What other work, in our field or in another, current or historical, has had the most substantial influence on your own?

26. By what criteria would you like people to judge your work? Do you think that *all* work in representation should be judged in this way, or does this apply in some specific way to your own project? If the latter, what are the characteristics of your system that

determine that these evaluation criteria are the relevant ones? (I.e. what would another system have to have in common with yours to deserve the same kind of evaluation?)

Section C -- Foundations

First a few questions about the overall nature of the enterprise:

27. The phrase "the representation of knowledge" can be taken to mean various different things. For purposes of discussion, we assume that:
    a: In building or using a representation system, one is supposed to "represent" something. In addition, this representation will somehow involve computational systems or ideas.
    b: A *symbol* is something which stands for something else. A *representation* is a particular kind of symbol, in which the structure of the symbol itself is perceived to correspond in some way with the structure of the thing that the symbol stands for. (Thus a painting of a farm might be a representation, because parts of the painting could be set in correspondence with parts of the actual farm.)
    c: One can (at least informally) distinguish among:
        P: the world
        Q: the active process of an intelligent person thinking about the world
        R: the language a person uses in speaking or writing about the world
        S: an abstract thing called "knowledge about the world"
    d: In any computational system, one can (at least informally) distinguish among:
        X: a set of data structures
        Y: a program or interpreter which manipulates these data structures
        Z: the overall active behaviour or process resulting from the running of Y over X.

    Given this breakdown, between what two things do you envisage the "representation" relationship? For example, do you think that X is meant to represent P (or R or S)? Or is Z supposed to represent Q? Both? Neither? ... If both, do you think the two goals could potentially come into conflict? Additionally: do you have a more precise characterization of the representation relationship that you use?

28. It is unclear to what extent the study of natural human language relates to research in knowledge representation. Many terms which arose first in the study of natural language (such as "proposition", "description", etc.) crop up in discussions of representation; this might be because these linguistic notions are necessarily central, or it might be because they are the best way we have of talking about some set of issues. More specifically:
    a: If your goal is to construct computer programs able to converse in natural language, then the study of language would obviously be of central concern.
    b: If your goal is more aptly described as constructing computer programs able to model intelligent thought processes, then the study of natural language is presumably relevant only to the extent that it sheds light on the underlying thought processes.

To the extent that they can be distinguished, would you characterize your inquiry as primarily an investigation into the structure of *language*, or more as an investigation into the structure of *thought*? If the latter (or if it includes the latter), how important do you feel the study of language is to your work? If it is relevant, do you find it requisite to do that study yourself, or are you able to base your work on the linguistic inquiries of others?

29. (Another question about language:) Three assumptions:
    a: The successful result of a study of natural language syntax will consist in the identification of a number of syntactic categories, and rules as to how words or structures in these categories may combine and interact.
    b: Similarly, the successful result of knowledge representation research will consist in the identification of a number of representation "categories", and rules as to how structures in these categories are allowed to combine and interact.
    c: In order to construct a program able to converse in natural language, one will need routines to translate from the "syntactic" accounts of natural language to the "internal" or "meaning" descriptions in terms of representation structures.

    Question: Do you think this translation will be specifiable solely in terms of syntactic and representation *categories*, or do you think that the translation will require specific reference to the individual words and terms to be translated? Put another way, do you think there could be found a "type-type" correspondence between syntactic and representational structures, or do you think that the mapping between the two will have to be specified at the "token-token" level? (For more discussion of similar issues see hypothesis 4 in part II.)

30. An extended analogy: Consider a performance of a musical composition such as a piano concerto. There are numerous ingredients which contribute to the total performance: the composition itself, the piano, the style in which the music was written, etc. We can set up an analogy between the many components of a musical performance, and those of a knowledge-based AI program. The musical composition, for example, can be compared with the complete computer program. In both cases this totality -- composition or program -- is influenced by its many constituent elements.

    Musical scales and modes can be compared with the primitive elements out of which the representation structures will be built: the one consists of a basic palette of available notes; the other a set of concepts, nodes, or relationships. Note that scales and modes have histories: the well-tempered scale, which is used almost universally in Western music (which divides the octave into twelve geometrically equivalent intervals) was invented only a few centuries ago. Similarly, modes -- that is, the selections of notes of a scale used as the primary melodic base of a composition -- are particular to different cultures (major and minor being the most common in Western classical music). One can imagine music research programs to develop new scales and modes.

The piano can be compared with the underlying computer language or hardware in terms of which program is implemented. The design and construction of pianos has its own history and evolution; grand pianos, for example, have key action (the arrangement of levers and hammers intermediating between the key and the string) that is superior to that used on upright pianos. One might compare this with the design of implementation techniques such as hash-coding or clever indexing schemes.

In music there are also various abstract forms and building blocks out of which larger compositions are constructed. It is because two compositions share similar building blocks that one thinks of them as being in a similar style. Some of the components are local, such as the arpeggio (a series of successively ascending notes), the cadence (a progression of harmony structure), or the trill. There are also broader organizational principles, such as sonata form, styles of orchestration, or kinds of instrumental group. In addition, many kinds of music (particularly popular) are distinguished by their rhythmic style. All these various ingredients can be compared to various building blocks out of which programs are built, such as general purpose subroutines, "packages" for performing some kind of computation, and styles of programming (such as the use of tail-recursion).

There are of course interactions at all levels: the music a composer will compose is constrained by the scale and style he adopts; the piano design is affected by the scale (the piano and the well-tempered scale, in fact, developed simultaneously). Some of the interactions are strong, but some are much more tenuous: the kind of keyboard action on a piano and the lyrical quality of the melody played on it affect each other only very indirectly. And again, comparable interactions, of similar flavour, can be found among the different facets of computational systems.

Nowhere in this discussion have we mentioned the role played by a knowledge representation system in an AI program. There are various alternative views that come to mind: you might, for example, compare the enterprise of designing knowledge representation systems or languages to one or more of:

a: designing pianos;
b: discovering scales or modes;
c: developing repertoires of melodic or rhythmic ingredients.

How do you think the role of knowledge representation research is best characterized, in terms of this musical analogy?

Two questions about semantics:

31. One can characterize:
    a: a *calculus* as a set of formal structures, and a set of conventions for manipulating them (cf. McDermott's "notational engineering").
    b: a theory of (denotational) *semantics* as a theory of how the constituent structures of a calculus correspond to objects or relationships in the world. (Note that this notion of semantics is that used by mathematicians and logicians; it is not the semantics of linguists.)
    c: a *logic* as an integration of a calculus with a corresponding theory of semantics.

According to this simple account, if you were developing only a theory of semantics, you would have to assume some calculus; if you were working only on a calculus, you would either ignore, or assume, a theory of semantics. If you were desiging a logic, however, you would have to be concerned with formulating both a calculus and a corresponding theory of semantics. Given this breakdown, would you characterize your project more as developing a calculus, a theory of semantics, or a logic?

32. We see in the computer science and philosophy literature references to various conceptions of semantics: denotational, operational, relational, and procedural, to name just a few. In addition, the linguistic tradition has its own conception of semantics. Which of these notions, if any, are of relevance to your work? Why or why not?

And four questions about logic:

(Note: By the phrase "*predicate logic*", as used in this questionnaire, we mean the logic of predicates and objects roughly as currently conceived by philosophers and mathematicians. This would include quantificational and perhaps modal logic, as well as the propositional calculus, and would also include both its traditional semantics and syntax (if not its traditional notation). What we do *not* mean is logic in as general a sense as is suggested in question 31 above.)

33. Quine has said that the ontological claims (assumptions about what exists in the world) made by predicate logic are merely that there are objects and relationships between them. Some representation schemes seem to assume the existence of other things, such as classes, sets, prototypical objects, or concepts. What ontological claims does your system make about the world?

34. Suppose we distinguish two (extreme) camps:
    a: One (represented perhaps by classical predicate logic) where the focus is on the formalization of statements about what is true, and on determining what statements can be taken to follow from others, without specific concern for (formally representing) the processes typically used by people in reasoning with such statements.
    b: The other (represented by most computer languages) where the focus is on the formalization of procedures to manipulate symbols, without specific concern for the meaning of those symbols.

    ... you see your work as being in one, both, or neither of these camps?

35. Many people working in representation are involved in designing new representation languages -- evidence, presumably, that they find predicate logic inadequate in one or more ways. Consider various alternative positions one might hold about the relationship between the new languages and predicate logic:
    a: Predicate logic is adequate, notationally and semantically: the hard problems in representation revolve around questions of what to say, not how to say it. Designing new languages, therefore, is of no particular interest.
    b: Predicate logic, while adequate for expressing a certain range of phenomena, is incapable of expressing some other kind or range of knowledge. New languages are needed to extend this expressive power.

c: Predicate logic, while adequate in some theoretical sense to express anything one might want to express, is organized or formulated in such a way that certain generalizations or constraints are not forthcoming in that formalism. The point of designing new languages is to find organizations which make these other generalizations self-evident.

d: Predicate logic considers only certain aspects of knowledge, and ignores others (such as memory organization and access). Consequently the importance of designing new languages lies in their approach to these other dimensions of knowledge representation, not in their having a new approach to issues with which predicate logic is concerned.

e: Predicate logic is founded on certain deep theoretical assumptions which are fundamentally wrong or inadequate. Consequently, no expression in predicate logic could be suitable.

Do you feel that predicate logic is inappropriate or inadequate as a representation system? If so, why? Do any of the positions given above reflect your view? If you find predicate logic inadequate, do you find it more inadequate as a calculus, or inadequate semantically (cf. question 31 above)?

36. One sometimes sees translations of new representation languages back into predicate logic. Whether you think such translations are appropriate probably depends on your answer to the previous question. In particular (except see also hypothesis 7 in part II), with each of the five positions set forth above, it would seem that we could associate the following corresponding position regarding translation:

a': Translation is a good idea, in that it captures what is said in the new languages, and provides a common ground for detailed comparisons.

b': To the extent that a new language succeeds in enabling the expression of this new range of knowledge, translation is impossible. However for the types of knowledge with which logic is concerned, translations would be adequate.

c': Translation is of course possible, but the generalizations and constraints will be hard or impossible to express in the translated version, and would certainly never have been discovered in that formulation.

d': Translation, in some sense, would be possible, but it would completely miss the point. It would be as if you spent a long time working out how to play a piano concerto on the classical guitar, and someone pointed out that your arrangement was no different from the piano version because it consisted of the same notes.

e': Translation is impossible, because of the lack of resonance between the two schemes. (This would seemingly be true particularly if one rejected the semantics of traditional logic.)

Do you feel that translations of new languages into predicate logic are capable of capturing what is essential about them? Which of the opinions above comes closest to expressing your position?

37. The previous two questions look at relationships between new representation languages and predicate logic. More generally, one would like to be able to compare any two representation schemes; to do this, one needs a sense of all the dimensions along which two representation languages could differ. Which dimensions do you feel are relevant in the design and comparison of representation languages?

Some questions more directly concerned with specific issues in representation and in computation:

38. (The issues raised in this question are explored further in hypothesis 9 of part II.) Many people, in writing about representation systems (particularly when discussing "frames" and "slots"), use the word "value". For example, one hears of "filling a slot with a value". However it is not clear what this term "value" means with respect to representation structures. Some observations:
    a: In computer science, we speak of expressions as having "values", and refer to a process called "evaluation" which, given an expression, returns its value. For example, the expression "3 + 4" would be said to have 7 as its value.
    b: We have in language the concepts of *description* (such as "the banjo my uncle used to play", and "my best friend in third grade") and of *name* (such as "J. S. Bach" and "Colette").
    c: Descriptions and names are said to have *referents*, but they do not in any obvious sense have *values*. (The referent of the name "J. S. Bach", for example, is a man who was born in the 17th century.)
    d: Since the referents of most ordinary descriptions are external to any computational system (there are no banjos or musicians in my PDP-10) there *cannot be* a computational process which takes descriptions and returns their referents.

    Is the notion of value, or the concept of evaluation (or anything analagous to it, built for example on the notion of referent) of any relevance in your system? If so, what is the notion, and how do you define it?

39. (Continuing question 38.) Another possible stand to take on evaluation and reference is the following:
    a: We can define two descriptions or names to be *co-referential* if they have the same referent. Thus the name "J. S. Bach" and the description "the famous German composer who lived from 1685 to 1750" are co-referential.
    b: We can then define a process which, given a description, is supposed to return a name which is co-referential with the given description.

    Does this, or anything similar to it, seem to you a useful move? If so, do you think it can be adequately defined and usefully controlled?

40. Some current languages (FRL, KRL, predicate logic, etc.) use a lexical syntax -- that is, their structures are presented in terms of things that look roughly like words and punctuation marks, assembled together according to certain rules of concatenation. (You could imagine typing these structures on a typewriter, for example.) Some other languages (many versions of semantic nets, KLONE, etc.) use a primarily graphical notation, at least for pedagogical purposes. Which kind of syntax is used in your system? Do you think there is any substantial difference in emphasis (some kind of Whorfian effect) that attaches to this difference and which, therefore, makes it important? If so, what do you see to be the essence of the distinction?

41. (A question on co-reference and projection.) As suggested in question 39, one possible account of what it means to fill in a piece of representation structure with a "value" is that you fill it with a co-referential name or description. For example, one might say that

(the value of) the name of Bilbo's friend the wizard is the string "Gandalf" (i.e. the 7-letter string with first letter "G", second letter "a", etc.) or that the (value of the) grade I got on my arithmetic test was 83. However, it can be argued that it is not true that the wizard's name *is* the string "Gandalf", but rather, that his name is *spelled* that way. Similarly, one can argue that the grade I got was not the number 83, but was somehow *symbolized* (or *expressed*) by that number.

David Levy has recently suggested the following terminology: that the string "Gandalf" is the *projection* of a name into the "space" or "world" of strings (he uses the term *"description system"*). Similarly, the number 83 would be seen as the *projection* of my grade into the number system.

First: Do you think your own name *is* some string, or do you think it is *spelled* as some string? Secondly: In your system, do you make use of any concept corresponding to this notion of projection? Do you think such a concept is coherent? If so, do you think that there is any straightforward "non-projected" co-reference, or do you think every relationship we call "co-reference" is really some kind of projection between different description systems?

42. A characteristic shared by many of the new representation languages is that of being "object-oriented": the declarative structures are organized around formal representations of objects. One can identify three primitive conceptual relationships typically employed in systems of this sort:
    a: The relationship between one individual and a "less specific" individual or class. Systems which embrace the notion of "classes" tend to use two forms of this relationship -- one of a class being a *subset* of another, and the other of an individual being an *element* of a class. Systems which opt instead for the notion of concept or prototype tend to use various flavours of a relationship typically called "IS-A".
    b: The relationship between one individual and another, where the first is a *component* or *part* of the second.
    c: The relationship between two representation structures, both of which refer to the same object. This (as suggested in question 39 above) is often called *co-reference*, and is indicated either with an explicit link, or by use of appropriately bound names.

    (Note that in the first two cases, the relationship is often given a different name depending on which end it is viewed from.) If your language is obejct-oriented, do you use varieties of these three relationships? If so, do you discriminate more finely, dividing each of them into more specific classes? (See the following question for a possible reason why one might want to do this.) Are there other primitive relationships in your system that do not fit into these categories? If so, what are they?

43. Most current representation schemes seem to deal extensively with the concepts of "abstraction" and "generalization". Furthermore, these two terms are often used interchangeably. It is not clear, however, that to do so makes sense. In fact, for purposes of discussion, let us characterize them differently, as follows:
    a: ***Abstraction*** is a relationship between *individuals*, one of which has been "abstracted away" from some dimension along which the other exists. For

example, suppose the car I drive is a Volvo PV544. Clearly my car is an individual. There is also another, more abstract, Volvo PV544, which is equally an individual. This is a car that was introduced in 1957, was discontinued sometime in the 1960's, and accelerates less well than the recently introduced (abstract) Porsche 928. Using our terminology, we would say that the abstract Volvo PV544 is an *abstraction* of my car. Similarly, one would say that the Wednesday of "Wednesday is Prince Spaghetti day" is an abstraction of the Wednesday of "on Wednesday I lost my watch"; that the pneumonia of "pneumonia can be cured with antibiotics" is an abstraction of the pneumonia of "my pneumonia lasted 3 weeks".

b: *Generalization* is a relationship between *concepts*, one of which is less-well specified than the other, and which can therefore be used in describing a wider range of objects. For example, if I describe my car both as "a Volvo" and as "a car", then the concept referred to in the second description is a generalization of the concept referred to in the first. Similarly, the concept *animal* is a generalization of the concept *grizzly bear*.

Do you agree with these characterizations? In particular, do you think the two notions are fundamentally distinct, or do you think they are (or can be defined in terms of) the same notion? If you think they are different, does your system deal with one of them? with both? Even if distinct, they seem related; what do you think is the relationship between them?

Also: Do you think there are other relationships of this sort -- relationships that hold between two things, one of which can roughly be described as being "less specific" than the other?

Several questions about theories, programs, and models:

44. Setting aside, for a moment, issues that deal specifically with representation, a question about the general role of computer programs in AI theory construction: People talk informally about "implementing theories", but it isn't clear what they mean. Suppose you have theory T which claims to explain behaviour B, and that you have implemented computer program P. Several possible relationships suggest themselves:

    a: (P is a *theory*.) T says something; P is a formal representation of T.

    b: (P is a *model*.) T states that there is a correspondence between the behaviour of P and behaviour B.

    c: (P is a *model explained by T*.) T states that behaviour B can be explained in terms of certain principles or mechanisms M. P is meant to be an inspectable example, designed both to exhibit behaviour B and to be explained by mechanisms M.

    It would seem that to demonstrate (a), one would need to show that there is a structural isomorphism between T and P. (If this is coherent, it is certainly radical, since it would imply that theories themselves are active. However, see hypothesis 8 in part II.) To demonstrate (b), one would have to show only that P effects behaviour B. To demonstrate (c), one would have both to show that P effects B, and also that P is explained by principles M. Question: Do you take (a), (b), or (c) to be the best account of the role played by computer implementations in AI theory construction?

45. Another view, regarding the role of the computer in constructing AI theories, is that the computer should be viewed as an *active scratchpad*, from a practical point of view as essential, but from a theoretical standpoint as irrelevant, as the paper and pencil of a mathematician. By this account, no program P would appear in the presentation of theory T about behaviour B. Do you subscribe to this view?

46. Questions 44 and 45 do not distinguish between general AI theories of intelligence, and theories specifically about the representation of knowledge. Picking up the piano analogy again, one can identify two obvious positions one might hold:
    a: (Knowledge representation systems are *tools*.) Since the enterprise of building knowledge representation systems is like designing pianos, the possibilities in question 44 would apply not to representation systems, but only to theories (and programs) built on top of them. Representation systems, therefore, should be judged only by whether they facilitate the development of other higher-level theories and programs.
    b: (Knowledge representation systems are *ingredients*.) Since the enterprise of building knowledge representation systems is like developing scales, cadences, or arpeggios, the issues raised in questions 44 and 45 are relevant to representation theories directly.

    Does position (a) or position (b) better reflect your opinion?

47. As well as asking about what theories in representation are, we need also to inquire about how they can be tested. Some observations:
    a: One strategy in testing a theory (derived from the traditional sciences) is to identify what phenomena it would predict, and what phenomena it would exclude, and then to verify that the predicted ones indeed occur, and that the excluded ones do not.
    b: In linguistics this strategy often takes the following form: the grammars set out by proposed theories are tested to ensure that they reject so-called "bad" sentences (those with syntax unacceptable to native English speakers).
    c: The purpose, in making sure that theories exclude phenomena which do not occur, is to guard against the construction of overly general (and therefore content-free) theories.
    d: It seems hard to imagine a lucid and compelling demonstration of how a given thought, inconceivable to the human mind, is impossible to represent in a proposed representation scheme.

    In light of these observations, do you think there is any kind of theory-testing, analogous to the linguists' testing of bad sentences, that is appropriate in representation work? If so, in what does it consist? If not, how do you suggest testing representation theories? Specifically, do you feel that it is important to ensure against the construction of overly general representation schemes, or do you think that simply attaining coverage is a sufficient requirement?

And finally:

48. Some of the emerging cognitive science programs around this country see artificial intelligence as being allied with psychology, linguistics, and perhaps education, as well as computer science, but not as being so close to philosophy. This questionnaire, on the

other hand, reflects a belief that philosophical questions are relevant to AI. What do you see to be the connection between representation research and these other disciplines? What do you see to be the most important contributions we bring to these studies from our computational background?

49. Suppose that tomorrow you were appointed philosopher-king. What 5-year plan would you mandate for the representation community? (In a hundred words or less.)

Section D -- Specific Representation Issues

Even a casual survey of the representation literature quickly reveals several hundred terms that are used, more or less technically, in refering to various aspects of the discipline. There are notions as cosmic as *knowledge* and *concept*, and as specific as *descriptions of prototypical sequence elements*; as external as *an object* and as technical as *a trigger to be invoked when an ungrounded anchor is made primary*. And in between lie myriads of *frames*, *defaults*, and *structured objects*.

Rather than ask you to react to (or define) hundreds of words, we have assembled a list of issues that seem to be of current concern to the representation community. They are listed below, grouped informally into eight categories. In the space at the right of each line (on the answer sheets) put a check in the column(s) which you feel best describe the issue, according to the following scheme:

a: An issue we (as a field) adequately understand.
b: An issue you are currently investigating, or which you address in your system.
c: An issue as yet unsolved, and important to investigate.
d: An issue as yet unsolved, but not of central concern at the moment.
e: A false issue, in the sense that you feel that it is an issue that should not be worried about. (Some may feel this about a search for canonical representation forms, for example.)
f: A valid enough issue, but outside the proper domain of "representation".
g: A vacuous or vague piece of jargon -- a term of no lasting value.

(Notes: In sections A and B, the issue to be considered is the *representation of entities in each of the categories*. Also, feel free to split up, and respond individually to, the various terms in each group, if your estimation of their status differs.)

Ontology and epistemology, basic:

1: Discrete objects
2: Structured objects
3: Classes, kinds, and instances
4: Prototypical or stereotypical objects
5: Individuals: criteriality, identification, individuation, and uniqueness
6: Individuals: abstractions, manifestations, and instances (see question 43)
7: Concepts
8: Ordinality and measure
9: Plurality (sets, collections, sequences, membership, partial orders, etc.)

10: Properties, qualities, and attributes
11: Quantification, quantificational scope, etc.
12: Continuity and substance (mass objects, stuff, etc.)

Ontology and epistemology, more complex:

13: Questions and commands
14: Goals and plans
15: Mechanisms: function, purpose, implementation, etc.
16: Causality, causal implication, etc.
17: Messages and linguistic expression
18: Spatial knowledge
19: Temporal knowledge
20: States, events, actions, change, etc.
21: Procedural knowledge (e.g. how to cook)
22: Processes
23: Situations and contexts
24: Abstract concepts derived from spatial metaphors, such as symmetry, parallelism, adjacency, etc.

Relationships between concepts:

25: Abstraction, generalization, and specialization
26: Multiple perspectives and multiple viewpoints
27: Description by comparison and by differentiation
28: Modification of descriptions (e.g. "*very* red")
29: Lambda abstraction
30: Mutually exclusive categories

Foundations and semantics:

31: Descriptions, representations, symbols, etc.
32: Expressions, pointers, names, identifiers, labels, etc.
33: Propositions, assertions, predications, relations, and functions
34: Reference, co-reference, and denotation
35: Intension and extension
36: Quotation and quoted descriptions
37: Truth and meaning

Logical properties and relationships:

38: Ambiguity
39: Declarative conditionals
40: Conjunction, disjunction, and negation
41: Contingent descriptions
42: Implications: temporal, causal, and inferential
43: Canonical forms and primitives
44: Completeness, logical adequacy, redundancy, and validity
45: Contradictions and consistency

Issues in deduction:

46: Evaluation, interpretation, simplification, and unification of descriptions
47: Induction, abduction, deduction, etc.
48: Inheritance, instantiation, and reasoning with defaults
49: Recognition
50: Pattern matching: residues, partial results, and sharing of results
51: Reasoning by analogy
52: Memory chunking, accessibility, and resource limitation
53: Indexing and retrieval
54: Depth of processing, partial computation, partial results, and failures
55: Searching, matching, and describing: frameworks in which to cast inferential processes

Knowledge treated as knowledge:

56: Assertion and denial of propositional content
57: Beliefs, hypotheses, and models of other people's knowledge and belief
58: Certainty, probability, and strength of belief
59: Meta-knowledge, meta-description, and meta-circular descriptions

Technical and formal notions:

60: Frames and slots
61: Nodes, links, and roles
62: Partitions, contexts, spaces, and other organizational mechanisms
63: Hierarchies, networks, and lattices
64: Agendas, priority queues, and process scheduling
65: Triggers and traps, servants and demons, and procedural attachment
66: Inference rules and production rules

Part II -- Some Non-Standard Hypotheses and Arguments

In this second part we have listed a set of hypotheses and arguments, each of which makes a strong claim about some aspect of knowledge representation. In each case, indicate whether you think the hypothesis or argument is obviously true, or obviously false; likely, or unlikely; generally questionable; or so vague as to be incoherent. (Remember that we are not necessarily advocating these hypotheses: in fact we have included some with which we disagree.)

In each case we present both a hypothesis and an argument. In some instances, the argument is essential: it may be as important as the conclusion which we draw from it, or it may even be necessary in order for the hypothesis to make sense. In others cases, the hypothesis by itself sums up the essence of what we are trying to set forth, and the argument is included only to convey at least a plausible sense of why one should take the hypothesis seriously. Cases of the latter sort are marked by having their "argument" sections introduced in parentheses.

In general, however, if you react differently to the hypothesis and to the argument (you believe one but not the other, for example), we would like you to indicate this explicitly. Similarly, when possible objections and secondary hypotheses are included, we ask you to comment briefly on whether you believe each of them.

1. Reference vs. Co-reference:

Hypothesis: Nothing that has been learned in philosophical theories of reference has any bearing on representation theories of co-reference.

Argument: Reference is a relationship which holds between a symbol and world; co-reference, a relationship between two symbols. While co-reference can ultimately be defined only in terms of reference (two co-referential symbols must by definition refer to the same thing, presumably), apart from this there is no relationship between the two notions.

2. Continuity and Stuff:

Hypothesis 1: It is impossible to construct a formal symbol representing any continuous object (according to the definition of representation given in question 27 of part I).

(Argument):
- a: Formal symbols are inherently discrete.
- b: The common world is inherently (if only partially) continuous.
- c: The task of knowledge representation is to construct symbols whose structure corresponds to the structure of that which, as symbols, they denote.

Objection: The hypothesis would be true only if one ignores the temporal dimension of denoting (see the next hypothesis), since the temporal dimension of a formal computational system is apparently (i.e. as far as the system itself can tell) continuous.

Hypothesis 2: To the extent that we can construct a formal symbol representing any continuous object, it must be the temporal dimension of the formal model which denotes the continuous dimension of the referent.

3. Temporal Denoting:

Hypothesis: The temporal structure of an active process (i.e. its structure in the temporal dimension) may denote some aspect of the structure of the object being symbolized.

(Argument): Computational systems are inherently active, and the temporal dimension is as basic a part of the structure of computational symbols as any other. While it is easiest to suppose that the temporal structure of a computational process should correspond to (denote) the temporal structure of the world being thought about, there is no a priori reason to assume that time should map directly to time; perhaps the mapping relationships could be more complex.

Example: In conceiving of a spiral staircase, the representation relationship holding between the "conceiver" and the staircase might be that the temporal aspect of the reasoning process corresponds somehow to the circular windings of the stairs. Or, in analysing a car engine, the temporal aspects of the analyser might structurally correspond in some way with the progress of the fuel through the engine.

4. Second Order Languages:

Definitions: We first need some definitions: (We will assume a representation language called REPL.)

a: By *"first order structure"* we refer to any particular terms which are individually referenced in the definition of the system's behaviour. Put another way, they are the primitive *vocabulary* which the system recognizes. For example, the first order structure of LISP would include things like "CAR", "LAMBDA", "QUOTE", and "TERPRI". The first order structure of REPL might include things like "OBJECT", "WHEN-IDENTIFIED", and "IS-A". The first order structure of predicate logic would include things like the truth-functional connectives, the quantifiers, and any modal operators. (See question 23 in part I.)

b: By *"second order structure"* we refer to the meta-structural categories in terms of which the system is defined. The second order structure would include all the meta-syntactic categories, types of operation, etc. For example, in LISP the second order structure would include the S-expression, the atom, applicative form, etc. The second order structure of REPL might include things like INDIVIDUAL-NODE, ISA-LINK, and CO-REFERENCE-POINTER. The second order structure of predicate logic would include things like the assertion, the proposition, and predicate-argument form.

Discussion: A few comments on the distinction:

a: First order structure corresponds more closely to the *lexical words* in terms of which the system is defined; second order structure corresponds more closely to its *grammatical categories.*

b: In general, names which identify second order structures are names of *types*, instances of which comprise the various structural pieces of the system. Names which identify first order structures *do not* name things which you can find in the structures of the system; however, what you do find are *occurences of the names* themselves.

c: For example, in LISP you find lots of S-expressions, but you don't find the atom "S-EXPRESSION"; hence "S-expression" is the name of part of LISP's second order structure. However, you do find the atom "LAMBDA", but you don't find any "lambdas" (although Guy Steele has recently argued that this should be changed). Hence "lambda" is part of LISP's first order structure.

Hypothesis: Ideally, a representation language should consist of *only*:

a: A well-defined structural definition phrased only in terms of second order structures, and

b: An interpretation procedure defined only with reference to second-order structures.

Comment: Note that the LISP 1.5 evaluation procedure does not satisfy this restriction, since it tests for atoms in function position being EQ to the constant, "LAMBDA".

(Argument): This argument assumes the position set forth in point (c) of question 44 in part I: that an "appropriate" AI computer program P should be explained by some theory T in terms of some mechanisms M. From this starting point, we can make the following observations:

a: Scientific theories -- for reasons of economy, generality, elegance, etc. -- usually state whatever they state in terms of *classes* of events, behaviours, objects, etc., not in terms of individuals. Gravity, for example, is a powerful theory because it applies to all objects, not simply to apples in Pisa. Similarly, powerful theories of inheritance or deduction will presumably apply to all concepts with particular structural properties (or something like that), rather than applying specifically to a few identified instances.

b: There are two ways in which this "law of generality" applies to representation systems. First, suppose you take representation systems themselves to be of theoretical interest (i.e. you believe they are ingredients, not tools). It follows that the most general explanation of the behaviour of a representation system should be in terms of its categories, not in terms of specific instances of its categories.

c: Secondly, suppose instead that you consider a representation language to be an implementational base, on top of which program P is implemented. Since P is to be explained in terms of mechanisms M, there is likely to be a correspondence between the second order structure of the language and the principles M. In this case, if the interpretation process were to make reference to the first order structure of the language, and the theory were to offer an explanation in terms of second order structural categories, it would be difficult to show that the theory legitimately accounted for the behaviour of the program. It might be possible, by inspecting the behaviour of the interpreter, to demonstrate the legitimacy, but it is hard to imagine *why* the interpreter should need to refer to first-order instances, if the theory is able to avoid doing so.

d: Finally: It is always possible, of course, for a theory to refer to first-order structure, even if the interpreter does not. This direction of imbalance creates no problems. However, since the ideal representation system should be able to support as powerful, and therefore as general, theories as possible, in its own definition it should make no reference to first-order structure.

5. Programming Considered Harmful -- Sense 1:

Hypothesis: Contrary to popular opinion, the worth of constructing a running computer program which "implements your theory" (see question 44 in part I) is roughly inversely proportional to the extent to which you understand it, and hence a bad idea.

(Argument):

a: It is always possible to construct ad hoc programs to model any specific given behaviour, without having reached any deep understanding of that behaviour.

b: Contructing computer programs requires paying attention to a myriad of details which consume time but are of no theoretical interest.

c: To the extent that any theoretical claim or insight is clear and well-understood, its structure should be comprehensible, and its worth apparent, independent of any implementation.

Objection: Indeed, the construction of a computer program is irrelevant in conveying the content or meaning of a theory. The reason to construct a computer implementation is not to see what the theory *says*, but to see whether it is *right*.

6. Programming Considered Harmful -- Sense 2:

Hypothesis: Imperative constructs in representation system programs can bear no theoretical weight.

Discussion: In order to construct an active computational model of theoretical interest, one ought not give the commands which effect the right behaviour, but *descriptions* of that relevant behaviour. This implies that the interpreter should be able to "understand" such descriptions, rather than simply react to straightforward directives. In more traditional computational terminology, one could say that in representation systems, data structures may be of theoretical interest, but code is not.

(Argument): Since any programs we construct are of value only to the extent that they say something about some world, we have to know how the formal objects of the program correspond to the objects of the world we are trying to explain. (This is presumably the *raison d'etre* of denotational semantics). Computer programs may consist of symbols, representations, descriptions, commands, etc. (whatever any of those might be). It is at least intuitively reasonable to imagine our having an account of how symbols, representations, and descriptions might correspond to the world. However it is not obvious how computational commands or imperatives could correspond. In particular:

a: It is not hard to see how the system's *behaviour*, resulting from the commands, might correspond to some behaviour in the world. The problem is rather that the correspondence between the commands themselves, and that behaviour, is obscure. In particular, you can think of a *description* of a computer process as analagous to a map of a road, whereas imperatives are analogous to road signs along the way. The problem is that the road signs are too *local*; they may say "turn right" or "turn left", but they don't convey an overall sense of the road. Each simply relates the next incremental section of the road to the previous section. In a similar manner, none of the imperatives of a computer program impart an overall sense of the process they are directing.

b: Constructing denotational theories of semantics for imperative constructs is therefore problematical. Some of the problems have to do with the locality: even if it were possible to account for some correspondence, such an account might be at a level too local to be interesting. Worse, there may simply not be any correspondence, at the level of the computational imperatives; all the structure of the process which corresponds to the structure being represented may be at a higher level than that of any given imperative command.

c: It may be objected that issues of locality and issues of imperatives are orthogonal -- that there are local and non-local descriptions, and local and non-local commands. However the fact is that as computational structures become more non-local, they tend to become more description-like and less command-like.

d: There are, however, problems more substantial than ones of locality. As suggested above, since the imperative constructs direct a process onto a new part of its path, such constructs make only *implicit* reference to the very process which they command. Consequently, there is no part of the construct directly corresponding to the active process under consideration. Hence there is no part of the construct which can bear the denotational relationship to the world.

7. Turing-equivalence Irrelevant:

Discussion: One often hears arguments that two computer languages are equivalent in expressive power, because they are both Turing machines, and (by Church's thesis) they can therefore both effect any behaviour that can be algorithmically specified. The hypothesis for discussion is that such "Turing-equivalence" is a vacuous notion when dealing with representation languages.

Hypothesis: In order to demonstrate that two representation languages are equivalent, one must show that they are not only behaviourally equivalent, but also structurally equivalent in such a way as to be equivalent semantically. One must demonstrate the equivalence, that is, of their referential apparatus. In particular, traditional notions of Turing-equivalence do not apply.

Argument: First, a premise:

a: In order to be a legitimate representation language, a language must include some account of how its data structures correspond to the world -- some theory of denotational semantics (that is, by question 31 in part I, it must be a full-fledged logic).

Second, suppose we are presented with representation languages A and B. Some observations:

b: The way that in practice one demonstrates the equivalence of two computer languages is to implement each one in the other.

c: Implementation does not in general preserve the identity of data structures. (For example: In the case at hand, suppose one started with A, implemented B in A, and then implemented a new A' in this B. The data structures in A' will not, in general, end up mapped exactly onto their equivalents in A.)

d: From (c) it follows that the normal way we implement languages in other languages does not preserve the denotational correspondences of the data structures with the world.

Implementation of one language in another, therefore, since it does not preserve denotational correspondences, does not constitute equivalence between representation languages. (This should not be surprising, since Turing-equivalence is based on behavioural, not structural, equivalence.)

8. Synthesis of Model and Theory:

Definitions: The hypothesis makes reference to the following two definitions:

a: A *theory* is made of sentences (or propositions), and has a structure that is different from the structure of the phenomenon it explains.

b: A *model* is not made of sentences (or propositions), and does have a structure that corresponds to the structure of the phenomenon it models.

Hypothesis: The definitions of model and theory given above are inappropriate. In particular, the boundary between the two notions should be made much looser, since computer programs can simultaneously exhibit features of both.

(Argument): Until the advent of the computer this distinction may have been sharp. However, since:

a: the data structures of computer programs can be so closely related to propositions; and

b: the data structures of computer programs can also be thought of as ingredients out of which working models can be constructed,

it follows that a computer program can simultaneously be both a model and a theory.

9. Evaluation as Reference-Finding:

Hypothesis: A coherent and simple concept can be defined which subsumes both the notion of *evaluation* of LISP and other applicative languages, and the notion of *reference-finding* (i.e. searching for, or identifying, the referent of a description or name) that is used in representation languages.

Discussion: First, we will argue that the evaluation procedure of LISP can be understood as a process which returns the *referent* of an expression, except in those cases where that referent is external to the system. In those cases, it returns some canonical symbol standing for the referent. Second, we will suggest that this is just the kind of "reference-finding" procedure which is needed in representation.

Note: The argument speaks of *numerals*. Whereas numbers are external and abstract entities, numerals (such as "1" and "5239.4") are *symbols*, of a particular canonical kind, which denote numbers.

Argument:

a: The common-sense notion of evaluation, as used in ordinary speech, and as regards mathematical expressions, is of a procedure which takes expressions and return numbers, not expressions denoting numbers. For example, note that in point (a) of question 38 in part I, the last sentence reads as follows: ... *the expression "(3 + 4)" would be said to have 7 as its value*.... In particular, it does *not* read: ... *the expression "(3 + 4)" would be said to have "7" as its value*....

b: LISP expressions can often be likened to descriptions. For example, the expressions "(CAR '(A B C))" and "(LIST 'A 'B 'C)" can be associated with the following descriptions: "the first element of the list (A B C)", and "the list consisting of the three elements A, B, and C". When a LISP system evaluates the original expressions, it returns what we would take to be the referents of the descriptions we have set in correspondence with them. This is possible because those referents are *internal to the system.*

c: If, however, we similarly view certain other LISP expressions as descriptions, we find that LISP does not return their referents. For example, when it evaluates the expressions "(+ 3 4)" and "(SQRT 144)", LISP returns *the numerals that denote the referents.* Similarly, certain atoms (such as T, NIL, and the numerals) "evaluate to themselves", not to abstract notions of truth or falsity, nor to abstract numbers.

d: There is thus a difference between our common-sense notion of evaluation, and the LISP notion of evaluation. The reasons for this difference are first that LISP *cannot* return certain referents (such as numbers), and second that you don't *want* those referents. Numerals are just as useful as numbers; they are, for example, what the arithmetic routines work with, and they are what we like to see printed out.

e: Just as LISP cannot return the common-sense value of a mathematical expression, so a representation system cannot return the referent of those descriptions and names which have external referents. On the other hand, just as there is no need for LISP to return actual numbers, so there is no need for representation systems to return the actual referents of names and descriptions, when those referents are external. What one needs, in cases like this, is some kind of symbol, analagous to a numeral, to stand for the referents of the common descriptions of representation systems.

In summary, one could divide descriptions (and expressions) into two classes: those with referents outside the system (we will call these ***first-level symbols***), and those with referents inside the system (i.e. ***meta-symbols***). An "evaluation" process could then be defined which:

a: Given a meta-symbol, would return its referent,

b: Given a base-layer symbol, would return some kind of "canonical" or special symbol standing in place of the referent.

Such a process would satisfy the description stated in the hypothesis.

Feedback

What other questions should have been included in this questionnaire? Do you feel that what you think is important about your project has been touched by these questions? Is there anything you would like to add?