AISB

SUMMER CONFERENCE
JULY 1974

University of Sussex

CONTENTS

Some Problems and Non-Problems in Representation Theory

Patrick J. Hayes

0. Introduction

The purpose of this paper is to give a brief survey of some general issues and problems in representing knowledge in AI programs. This general area I will call representation theory, following John McCarthy. Its boundaries are, like those of all interesting subjects, not crisply defined. It merges in one direction with programming language design, in another with philosophical logic, in another with epistemology, in another with robotics. Nevertheless, it is an increasingly important aspect of AI work. Since my main concern here is to draw attention to problems which seem to me to be difficult, and issues which seem to be important, this paper should be read as an appeal for help rather than a statement of achievements (and comments, criticisms and suggestions are welcome).

Inevitably, to believe that some issues are important, and some problems difficult, is to believe that others aren't. At the end of the paper I draw attention to some specific points of disagreement with other authors. It may be helpful, however, to point out immediately that my goals here are not philosophical, but technical. Some commentators on an earlier draft seemed to take it as an essay in philosophical analysis in the modern Oxford style. My aim rather is to substitute, for informal and apparently endless philosophical discussion, the precision of mathematics. (This aim is not achieved in this paper, I hasten to add, but is I hope brought nearer.) To emphasise this, I will, when introducing a technical word intended (ultimately) to have a precise meaning, underline it.

1. Semantics

There are many ways known of systematically representing knowledge in a sufficiently precise notation that it can be used in, or by, a computer program. I will refer generally to such a systematic representational method as a scheme. It is not a very good word, but one cannot say 'language' as that begs an important question (see section 2). Examples of schemes include logical calculi, some programming languages, the systematic use of data structures to depict a world (e.g. as in the early Shakey's use of an array as a room-map), musical notation, map making conventions, circuit diagrams, 'JCM Schemas', 'Conceptual Dependency' notation, 'Semantic Templates' (all in [21]), etc. A configuration is a particular expression in a scheme: an assertion, a program, a data structure, a score, a map, etc. Thus one might, formally, define a scheme to be a set of configurations.

All of these examples are formal in the sense that the question, whether a particular arrangement of marks is a well-formed configuration, always has a definite answer: there is a definite notion of well-formedness. Many ways which humans have of conveying meaning will not be allowed as schemes, for they fail this criterion: drawings, photographs, poems, conversational English, musical performances, TV pictures, etc. In brief, I wish to draw a distinction between (formal) schemes, in which knowledge can be stored and used by a program, and on the other hand, (informal)

scenes or perceptual situations requiring the deployment of knowledge for their successful interpretation.

I am aware of several philosophical problems in analysing this distinction further. As a rough-and-ready guide, schemes can be recognised by the fact that one can construct ill-formed 'configurations'. There is no such thing as an ill-formed photograph. Natural language is a borderline case, as are accurate line drawings of polyhedra.

Schemes are usually intended as vehicles for conveying meanings about some 'world' or environment. In order to be clear about this important topic, a scheme must have an associated semantic theory. A semantic theory is an account of the way or ways in which particular configurations of the scheme correspond to (i.e. have as their meanings), particular arrangements in the external world, i.e. the subject matter about which the scheme is intended to represent knowledge. Some of the schemes referred to above have very precise semantic theories, others have none (and seem to rejoice in this lack: see section 7 below), others (music, maps, circuit diagrams) have informal semantic theories which can be made precise by the approach outlined in section 2 below.

It is not at all fashionable in AI at present to give semantics for new representational schemes, and this is, I believe, a regrettable source of confusion and misunderstanding. Now, one cannot prove such an opinion, of course. One can point to other fields where syntactic confusion and proliferation of ad-hoc formalisms has been or is being replaced by the development of semantic insights: notably, philosophical logic and the design of programming languages. One can point to the way in which, in AI itself, elementary semantic ideas have been re-invented by various authors over the years (especially the Frege/Tarski notion of individuals and relations between them, which crops up with remarkable regularity [, ,]). And one can point to several important questions which simply cannot be answered without a semantic theory. Of these, the most urgent concern the equivalence or otherwise of different formalisms. Is there a difference in meaning between a conjunction of atomic predicate-calculus assertions and the corresponding semantic network? Is there anything which can be expressed in the notation of Merlin [16] which cannot be expressed in a logical notation? The answer to both these questions is yes, in fact: but without a semantic theory the questions cannot even be precisely formulated. Finally, discussion in the AI literature, on, for example, the different roles of deductive, inductive and analogical reasoning and the relative merits or demerits (either technical or philosophical) of various formalisms, is often ill-informed or at best vague due to a lack of a clear model theory for the systems under discussion.

Nothing so far has been an argument for any particular sort of semantic theory: for example, some kinds of 'intensional', 'operational', 'meaning-intentional' or 'procedural' semantics, may eventually enable the meanings of configurations in a scheme to be rigorously defined. However, as a matter of fact, the only mathematically precise account which I have seen of how a scheme can talk of entities outside of the computer, is the Tarskian model theory for first-order logic (but see section 2 below). I believe there are important reasons for going beyond this semantics, but many of the arguments in the AI literature against the use of predicate logic as a scheme are based on misunderstandings of one kind or another, especially the assumption that the use of predicate calculus necessarily involves the use of a general-purpose theorem-proving program. (See section 7 for more discussion.) To defend first-order logic is unfashionable: however, I do want to emphasise that it is the semantics of predicate logic

which I wish to preserve. I have no brief for the usual syntax: networks, for example, can be used as a syntactic device for expressing predicate calculus facts. Some other authors advocate rather the use of predicate calculus syntax either without semantics [19], or with an alien semantics imported from computational theory [6]. This is throwing out the baby and keeping the bathwater.

To insist on a semantic theory is not, of course, to insist that the expressions comprising a program's beliefs are accurate, i.e. that what they express about the world is in fact the case. (This common misunderstanding may be caused by the phrase "truth-recursion", which leads people to think that metamathematics guarantees infallibility.) Without a semantics, one cannot even say precisely what is being claimed about the world: that is the point.

It is important to emphasise that to regard a formalism 'simply' as a programming language: that is, a way of getting the machine to do what one wants, is to adopt a rather different point of view towards it. (Unless, that is, the semantics of the scheme are concerned with machines and what they do.) For example, many people argue that PLANNER is to be regarded 'simply' as a programming language which provides useful facilities for the sorts of programming one finds oneself involved in when writing AI programs. Much of the force of the criticism in [23] for example, is from this position. While this is a perfectly respectable point of view, it is different from the one which regards PLANNER as a scheme which refers to external worlds of, say, blocks. It is even different from the idea that PLANNER is a scheme which refers to problem-solving processes or the like. For the 'programming language' view encourages the user (for example), if he needs a new semantically primitive notion, like negation, to encode it - that is, to implement it - in PLANNER in some way. In terms of schemes this is a change of scheme, since the semantics have been enriched.

To put it extremely: the only difference, in this view, between (say) CONNIVER and (say) FORTRAN, is user convenience: for one could implement the one in the other. (I have heard precisely this view forcibly maintained by professional systems programmers). Hewitt characterises the essence of PLANNER in terms of schemas [13]. While this syntactic approach works up to a point, the relationships between programming languages are, I feel, greatly clarified by giving them natural semantics. The trivial universality which FORTRAN possesses can then be eliminated by the requirement that in embedding one language in another there is a corresponding embedding of the meanings of programs. "Implemented in", as a relation between languages, then ceases to be an embedding since the meaning of (say) THCONSE does not correspond to the meaning of the rather large piece of (say) FORTRAN which would be in the implementation (actually, several pieces scattered about the program but related by context.) The former has to do, presumably, with goals and facts and such things: the latter, probably, with arithmetic relationships between numbers which represent list structures in some way.

In saying all this, one must admit that there is much force in the position that it is too early in AI to settle on particular schemes with fixed semantics. According to this view, AI programs should be implemented using all possible programming skill and ingenuity and we should leave to the future the (perhaps rather arid) task of tidying-up. Much very good AI work has been done from this standpoint, and will probably continue to be done. I do not wish to give the impression of arguing against pragmatic expediency in writing advanced programs. But I do feel that it is not too early to investigate schemes with organised semantics, both on general grounds of scholarliness and because I believe that such schemes are

ultimately easier to use in programming.

2. Linguistic and direct representations

Several authors have drawn attention to a distinction between representations consisting of a description in some language and representations which are in some sense more direct models or pictures of the things represented. I first met this distinction in [1], and it has been more recently emphasised by Sloman [22]. It seems to be clearly important but I have met with surprising difficulty in trying to make the distinction precise.

One problem is to suitably define what is meant by a descriptive language, for we must not beg the question by being too restrictive in our definitions of language. Thus Sloman's emphasis on what he calls analogical representations is really a plea for the consideration of a wider class of languages than those in which the only semantic primitive is the application of a function to arguments (Sloman's term is 'Fregean' languages, like predicate calculus and PLANNER. Some authors seem to have interpreted Sloman as arguing against the use of descriptive representations [3], but this is a misunderstanding.)

Another problem is that a representation which appears to be a direct model at one level of analysis, may, upon enquiring further, be itself represented in a descriptive fashion, so that it becomes impossible to describe the overall representation as purely either one or the other. For example, a room may be directly represented by a 2-dimensional array of values which denote the occupants of various positions in the room: but this array may itself be implemented by the programming system as a list of triplets <i,j,a[i,j]>, i.e. by a sort of description. It seems essential, therefore, to use a notion of level of representation in attempting to make the distinction precise.

Third, any representation must also be a direct representation of something. For, the pattern of marks which is a configuration of the scheme, can convey meaning only by virtue of the fact that its parts are physically arranged in some definite way. This physical arrangement has to be a direct representation of (at least) the way in which meanings of some configurations are compounded into meanings of larger configurations.

Fourthly, the notion of direct representation seems to depend upon some similarity between the medium in which the representation is embedded, and the thing represented. Thus a map of a room is a direct representation of the spatial relationships (in the horizontal place) in the room, by virtue of the similarity between the 2-dimensional plane of the paper and the 2-dimensional plane of the floor of the room. The paper is a direct homomorph of the room: they are the same sort of structure (2-D Euclidean space), admitting the same sorts of operations (sliding, rotation, measurement), but the map is a simplification of the reality, in the sense that certain properties present in reality (colour, exact shapes, etc.) and certain relations (the third dimension, comparisons of value) are missing in the map. Another example is an ordered vector of items in a core store: here the medium is the address structure of the store, which is similar to the integers in respect of its ordering relationships, but not (for example) in respect of its cardinality (stores are finite).

Putting all this together, one arrives at the following general position. There are things called media in which one can construct certain

configurations of marks or symbols: that is, arrangements of marks in which relations exhibited directly in the medium hold between the marks. A language is defined (syntactically) by a set of 'primitive' symbols and a set of grammatical rules which define new configurations in terms of old ones. One gets the usual ideas of parsing. (It could be mathematically interesting to see how much of formal language theory can be generalised to this setting from the conventional 'string' case of 1-dimensional media. One can certainly define context-free, and context-sensitive grammars, but I am not so sure about finite-state, for example.) A model for such a language is provided by a set of entities acting as meanings of the primitive symbols; and, for each grammatical rule, a semantic rule which defines the meaning of the configuration in terms of the meanings of its parts. (One needs variables and variable-binding expressions also, so this account needs elaboration and qualification, but space does not permit a full discussion.) This, so far, is the usual Tarskian idea of a truth-recursion, generalised to this more general notion of language. But now, we also insist that each medium-defined relation used in constructing configurations corresponds to a similar relation in the meanings, and that the representation is a structural homomorph of the reality with respect to these relations. That is, the meanings of configurations must exist in a space which is similar to the representing medium, and the syntactic relations which are displayed directly by the symbol-configurations of the language, must mirror semantic relations of the corresponding kind. The directness of a direct representation lies in the nature of the relationship between the configurations and the reality they represent (it is a relation of homomorphism rather than denotation). A scheme is not direct because of any syntactic features (such as being 2-dimensional) of its schemes, or because of any special qualities (such as being continuous) of the worlds it describes.

It is possible to give formal grammars for simple maps, to emphasise how this account fits the facts, along the lines of Rosenfeld's isotonic grammars [18]. To emphasise again: map-making conventions are, in this view, a language, of which the maps are expressions. The relationship of these expressions to reality is that the primitive symbols denote features of a terrain in a way defined by the map key, and the positional relationships between symbols directly display corresponding relationships between the denoted features.

In electrical circuit diagrams, lines joining symbols denoting components directly denote, in their topological structure this time, the electrical connectivities in the actual circuit. Another example is provided by the simple narrative convention. In "He got up. He got dressed. He went out. He walked to the shop ... ", we understand a time-sequence which is directly denoted by the ordering of the (timeless) separate propositions. This convention is also used in programming languages and cartoon strips, with the same sort of semantics. A final example is provided by networks. A network is a configuration which is a relational structure. Web grammars are the appropriate parsing device. The most obvious way of giving this a semantics is by declaring that a model is any relational structure into which the network can be homomorphically embedded. According to this semantics, a network has the same meaning as the conjunction of predicate calculus atoms corresponding to the arcs of the network. (It is a straightforward exercise in system programming to convert a list of such atomic assertions into a network, represented in the core-store medium by using 'addresses' as the direct analog of 'is connected to', for efficient retrieval.) As we will see, however, one can give a rather different semantics to networks, which makes them more expressive in an important way.

A more complete and rigorous account of this will be published elsewhere. The major problem is to find a general precise characterisation of what is meant by "medium" and "similar". I am currently working on an algebraic account (in which a medium is a category), but it is not yet altogether satisfactory. (Suggestions are welcome.)

The importance of all this, apart from the intrinsic interest of the subject, seems to me to lie in three points. (1) It shows that direct representations are not incompatible with linguistic representations, and can be given a precise model theory along Tarskian lines (which supports Sloman's view in [22]). (2) It suggests ways in which efficient deductive systems may be generalised from work in computational logic. (3) The notion of 'medium' captures the idea of levels of representation mentioned earlier. For a medium may not be physically directly present, but may itself be represented by configurations in some quite other medium, as in the array example. Or again, consider a simulation language like SIMULA. This provides a medium consisting of processes and events and certain relations between them. This medium, taken in its own terms, gives a direct representation of time which is often extremely useful. But if one goes deeper, time is represented in a quite indirect way involving numerical descriptions and long chains of inference. This 'looking-deeper' means not treating SIMULA as a medium to be used to represent, but rather as a reality which is itself represented in some medium (say, FORTRAN or assembly language). The choice of primitive relationships defines both the medium and the level at which analysis will cease.

This shows, incidentally, that Sloman's arguments for the utility of analogical representations, based on the idea that they are somehow more efficient in use than Fregean representations, are fallacious. For an analogical representation may be embedded in a medium which is itself represented in a Fregean way in some other medium. Any discussion of efficiency must take into account the computational properties of the medium.

3. Exhaustiveness and plasticity

An important fact about schemes with Tarskian semantics is that a configuration in such a scheme is, in general, a partial description of the environment. It constrains the form of a satisfying world, but does not (in general) uniquely determine one. And even if it does uniquely determine a world (is categorical, in the technical term), this fact can only be determined by metamathematical analysis: there is no sense in which one can say in the scheme itself, "this is a complete description".

Now this means that one has the opportunity of adding information ad lib, further specifying the world. (Hence the idea of conjunction arises very naturally). The process of adding information can be arrested only by the whole configuration becoming inconsistent, i.e. making an assertion about the world which is so strong that no such world exists. Different schemes will have different particular notions of consistency, but this general outline follows from the abstract properties of the satisfaction relationship between configurations and worlds. This ability to accept new pieces of information and to gradually accumulate knowledge piecemeal is one of the most valuable aspects of Tarskian schemes. Thus, the idea of a 'knowledge base' of separate pieces of information, to which new pieces can be added freely without a need,in particular, to pay attention to control flow or other organisational matters, is very familiar and important.

This possibility of adding information is one aspect of a scheme's plasticity, i.e. the ease with which changes can be made to configurations in the scheme. Plasticity is essential for nontrivial learning, and for any system working on limited information in an uncertain world.

However, there are times when one does want to be able to make a claim of exhaustiveness in a representation. For example, we might want to represent that all the relations of a certain kind, between the entities represented in the configuration, are also represented in the configuration; or, that all the facts about some entity, which are in some sense relevant to some problem or task, are present in the configuration.

One important example of the need for this sort of assumption is the well-known frame problem. Consider a traditional description of the monkey-bananas problem, in natural English. How do you know there isn't a rope from the box, over two pulleys, and down to the bananas (so that as you move the box, the bananas ascend out of reach)?* Well, we assume that the simple description has given us all the relevant information to do with causal chains in the situation: we assume it is an exhaustive account of the machinery of the room. Much of the difficulty of the frame problem lies in the impossibility of expressing this assumption in the predicate calculus. (Using the causal-connection theory developed in [9], we could say there was no causal connection between the box and the bananas; but that is not strictly true: the monkey can throw one at the other, for example. In any case it is unsatisfactory as a general solution.)

(Parenthetically, I would like to take this opportunity of suggesting that we should stop talking about the frame problem. There are, it is now clear, several independent difficulties bound up in the normal formulation. One was just noted; another is the lack of a good representation of the way in which causal chains follow trajectories determined by mechanisms in the environment; another is the heuristic problem of organising inferences involving causality. The presence of state-variables in the language is not part of the problem, as some authors seem to have believed.)

Another, rather different, example of a claim of exhaustiveness is provided by the sort of analogy reasoning epitomised by Evan's well-known program, and formalised in the Merlin system [16]. This is normally regarded as essentially non-deductive reasoning, but it can be regarded as deductive reasoning from some rather strong hypotheses. Thus, suppose we decide that a certain collection of properties of an individual, taken together, constitutes an exhaustive description of it, from a certain 'point of view'. For example, we might say that a man was a mammal with a nose and feet. What could this mean? Well, it might mean that certain facts about men can be established by the use of these properties only: that is, an essentially proof-theoretic assertion. Now, with this meaning, if we replace the properties in the description with others (of the same 'type', in some sense: e.g. with corresponding sort structures in a multi-sorted logic), then corresponding facts can be established relative to the alternative properties. Thus, in the example of [16], if a pig is a mammal with a snout and trotters, then we can regard a pig as a man with a snout for a nose and trotters for feet. The existence of the 'analogy' follows from the (presumed) sufficiency of the list of properties. It follows deductively from the claims expressed in the putatively exhaustive descriptions of men and pigs.

---

*This example due to Alan Newell

This account of analogy (which is related to Kling's ideas) suggests natural explanation of (for example) the breakdown of an analogy (the claim of exhaustiveness fails: e.g. some property of men needs other hypotheses than those of noses and feet), and naturally relates 'analogical' and 'deductive' reasoning.

Now, there is a way in which a direct representation can be considered to be exhaustive, by a slight alteration to the semantic rules. We may insist that the medium-defined relations of a configuration completely mirror the corresponding relations in the reality: that is, that a medium-defined relation holds between subconfigurations if and only if the corresponding relation holds in the world between the entities denoted by the subconfigurations. Let us call such a representation, strongly direct.

For example, a map is strongly direct in this sense: all the 2-dimensional spatial relationships which hold between towns, rivers, etc. also hold in the map between the symbols denoting them. (They are also, often, exhaustive in a stronger sense; that all the entities (towns, rivers) present in the reality are denoted by symbols in the map. Thus we say, of a map with a river missing, that it is wrong, not just incomplete. It misleads us because we assume that if a river isn't marked, it isn't there.)

An example of a direct representation which isn't strongly direct is provided by networks: a relation may well not be displayed in the graph. However, we can also use networks as a strongly direct representation, if we consider the medium to be the algebra of relational structures with a given signature. Thus we would insist that either all or none of the instances of a certain relation are displayed in the network. A family tree is a strongly direct representation in this sense, relative to the relationships 'child of' and 'married'. With this semantics, (which can be specified algebraically) a network is no longer equivalent in meaning to the simple conjunction of the atomic facts represented in it. (If we call this conjunction C, it is equivalent to C with the added rule: if C $\not\vdash \emptyset$ then $\neg\emptyset$, for any atom $\emptyset$ in the appropriate vocabulary.) Winston's use of networks to describe concepts [26] seems to be closer to this latter semantics than to the former one, for example.

In unpublished work at Stanford, Arthur Thomas is developing a different approach to combining exhaustiveness with a Tarskian semantics, based on Hintikka's 'model sets'.

Strongly direct representations are less plastic than direct/Tarskian representations, in that information cannot be accumulated piecemeal in them. To add information to a strongly direct representation is to alter the information expressed by it. Alterations, as opposed to mere additions, raise problems of their own.

The trouble with alterations is that the information being altered may have been used earlier as a premis in a deduction of some kind. Thus, other pieces of information which obtain their support in some sense, from the altered information, are now endangered, and should probably be re-examined. This seems to require the system to keep an explicit record of how it formed its beliefs: a history of its own thinking. And this seems prohibitively expensive (of either space or time: one could recompute rather than store), due to exponential factors in the amount of information required.

Under some circumstances, it may be possible to re-evaluate a belief on criteria independent from its original derivation, as for example in adjusting the fit of lines to a gray-level picture (this observation due to Aaron Sloman), but in general I do not think one can avoid the problem.

This dilemma seems insoluble. There must be a clever series of compromises which steer us between its horns, but I don't know of any work in this direction.

More far-reaching alterations to a representation which one can envisage include changes to the basic ontology, to the sorts of entity to which it refers. The introduction of substances into a scheme oriented towards describing individuals is such a change, for example (see section 6). Minsky and Papert [15] give another rather simpler example: the change from a two-place relation of support between objects to a support relation between an object and a collection of objects, needed to describe e.g. an archway or a table. As they remark, this alteration seems to require a complete rebuilding of all knowledge about support, for the actual logical grammar of the assertions has changed. However, in this and similar cases one can see the general outlines of how it might be done. The fundamental step is to introduce the new notion of support as a new primitive idea (this is the really 'creative' act), and then define the old notion in terms of the new one, i.e. regard the old concept henceforth as an abbreviation for its definition in terms of the new one. In the example, support (a,b) would be defined as support (a,{b}). This preserves the old theory of support as a special case of a new, more general, theory (which is yet to be defined). There is, however, a strong constraint on the new theory, viz. that it'explains' the old theory. Thus, statements of the new theory which translate statements of the old theory must be derivable (in the new theory).

This corresponds to the idea that the alteration is somehow a refinement of, or an improvement upon, the former representation. A similar change, but in which the new concept completely replaced the older concept, which was rejected as wrong or unusable, could not be handled this way.

This whole issue of plasticity in representation is important not only for learning, but also for everyday program development reasons, and for debugging. For we must be able to modify and improve the representations of knowledge in the programs we write, and this is often far from easy.

4. Evidential Reasoning

There is a continual need, especially in perception, to represent information concerned with one belief being evidence for another. It seems clear that one needs to make reasonings concerning such matters explicit so that they can be properly related to other reasonings, and can be adjusted in the light of experience (see section 3). The problem is how to adequately express the notion of one knowledge-fragment (or collection of fragments) being 'good evidence' for another.

There seem to be several notions of good evidence, but all can be put into a common framework: A is good evidence for B (under assumption Th, say) if the conjunction (A & not B) is somehow unlikly or implausible (or: if this follows from Th). Thus, for example, if A entails B then A is very good evidence for B, for then (A & not B) is impossible. In Guzman's work [8] back-to-back 'T's are good evidence for occlusion of

one body by another, since the former without the latter is an unlikely coincidence. In a world where lines of bricks were common, back-to back 'T's would be weaker evidence since the conjunction of such an observation with the hypothesis of a single occluded body would be less implausible: the possibility of a line of bricks being occluded would be an alternative explanation of the evidence.

This sort of observation suggests an account of 'plausible' as follows: (A & not B) is implausible if B entails A (occluded body entails back-to-back 'T's) and no other B of the suitable sort (e.g. no other hypothesis about physical arrangements of bodies) entails A. If there are several such explanations of A then A is evidence that one of them holds, but it doesn't distinguish which one. This decision has to be made on some other basis, for example by the use of Baye theorem in a probabilistic scheme, or by choosing the simplest hypothesis or the one most compatible with other entrenched beliefs.

An important problem is how to discover the collection of possible or likely explanations. (This point was emphasised to me by Aaron Sloman). How many ways can back-to-back 'T's arise? I can think of three, and am pretty convinced there aren't any more; but I have no idea where that conviction comes from, or how I would prove it. The 'theory' of lighting and perspective which is welded into Waltz's program has this nice exhaustive character, expressed in effect as a collection of explicit disjunctions. This works up to a point, but how could a program derive these lists from a description of, for example, the lighting conditions and geometry of the scene?

Involving the background theory of lighting, etc., in this way is not just of academic interest. A vision system which could make hypotheses about the lighting conditions, the sorts of reflectivity in the scene, etc. would find it necessary to be explicit about the role of such assumptions in interpreting pictorial phenomena. Thus we might have: if there is strong unidirectional lighting then shadows have sharp edges and are dark; so if this is the corner of a shadow then it will have a dark interior: $\overline{Th} \supset (B \supset A)$; from which we may use corners with sharp edges and dark interiors as evidence for shadows. Reasonings like this will be essential in any system with the ability to percieve a range of scenes. (Similar remarks apply to other perceptual situations, e.g. understanding speech, handwriting, children's stories.)

5. Control

A system which makes inferences to generate new facts must control its inference-making capabilities in some way. This control itself requires the storing and using, by the system, of information about the deductive process. That is: the system must represent and use knowledge about its own deductive behaviour.

In conventional programming languages this information is sometimes represented implicitly in, for example, the ordering of statements in the body of a program (which is a strongly direct representation of the time-order of control flow, provided jumps are forbidden) and sometimes explicitly in, for example, the correspondence in names which relates procedure calls to their corresponding procedure bodies. In PLANNER-like languages, the latter representation breaks down since 'procedures' are called not by name but by pattern matching, and is replaced by the more flexible device of advice lists. The ordering information is still represented implicitly, however.

Now, this metadeductive information needs to be made explicit and separated from the factual information represented in the scheme, for reasons of semantic clarity, plasticity and deductive power. For example, the residue of PLANNER upon separating out control information is a logic which resembles intuitionist predicate calculus . Results like this are important: they give us an inkling of how a semantic theory might be put together. (Unfortunately, intuitionist logic itself has a rather murky semantics.) The control information which can be represented in PLANNER is rather limited, as the CONNIVER authors emphasise [23]. Their solution, to give the user access to the implementation primitives of PLANNER, is however, something of a retrograde step (what are CONNIVER's semantics?), although pragmatically useful and important in the short term. A better solution is to give the user access to a meaningful set of primitive control abilities in an explicit representational scheme concerned with deductive control. This is the basic idea of the GOLUX project now underway at Essex [11].

The problem is to find a good set of control primitives. What is control? One answer to this is to pick on a fixed mechanism (the interpreter) associated with the language, and to relate control to this mechanism in, more or less, the way an order code relates to an actual computer. But this tends to be inflexible and arbitrary. The GOLUX answer is that control is a description of the behaviour of the interpreter. The exact nature of the interpreter is not defined, only that it constructs proofs according to some predefined structural rules. The descriptions in control assertions constrain its behaviour more or less tightly. It is, I believe, important that control information be represented in a scheme compatible with the scheme used for 'factual' information, so that control can be involved in inferences, added to, and changed.

Control primitives in GOLUX include predicates on, and relations between, partly constructed proofs in the search space; descriptions of collections of assertions; and primitives which describe temporal relations between events such as the achievement of a goal (e.g. the construction of a proof). The major source of difficulty is the tension between the expressive power of these primitives and their implementability: it is important that they be sufficiently simple that their truth can be rapidly tested against the actual state.

GOLUX is based on recent ideas in computational logic [10,12]. Other authors have also recently emphasised that computational logic provides a powerful theoretical framework for problem-solving and computational processes [14,28,19], although we are not in complete agreement as to which is the best framework.

A common area of difficulty both here and in evidential reasoning is to get a good notion of a 'theory': an organised body of knowledge about some subject-area.

6. Substances, Parts and Assemblies

Every representational scheme known to me is based ultimately, like predicate calculus, on the idea of separate individual entities and relations between them.

But our introspective world-picture also has quite different 'stuff', viz. substances: water, clay, snow, steel, wood. Linguistically, these are meanings of mass terms. Substances are fundamentally very different from individuals, and I know of no scheme which seems capable of satisfactorily handling them. I became aware of this problem from reading

Davidson [5].

We often speak as though substances were individuals having properties and relations one to another and to more conventional individuals: steel is dense, blood is thicker than water, his head is made of wood. The relation "made of" seems particularly important. But appearances are deceptive.

Does 'water is wet' mean the same as 'all samples of water are wet'? I think it does: we certainly want to be able to infer from 'water is wet', that 'this sample of water is wet'. This suggests at first sight that we should treat pieces of stuff as individuals, which seems fairly acceptable. But these individuals are also rather strange, especially for fluids. If you put together two pieces of water you get one piece, not two: we have to speak of quantity (of stuff) before we can use arithmetic. (It is significant that, as Piaget has shown, children properly understand the concept of quantity only at quite a late stage of development.) Moreover, we should distinguish properties which a piece of stuff has by virtue of its being a piece (size, shape), from those which it has by virtue of its being made of stuff (density, hardness, rigidity): for the former, but not the latter, can be easily altered by physical manipulations. It really seems that we cannot get away from substances no matter how hard we try.

Let me emphasise that this problem is not a by-product of a nominlist philosophical position. I have no objections to platonic, abstract, non-physical individuals. That's not the difficulty. The difficulty is 'individuals' which appear and disappear, or merge one with another, at the slightest provocation: for they play havoc with the model theory.

This seems to me to be one of the most difficult problems in representation theory at present. The only way I can imagine handling substances is by regarding each substance as a (special sort of) individual, to which such properties as hardness, density, etc. are attributed. These individuals can be regarded as platonic ideals, or alternatively as the physical totality of all samples of the substance: you can take your nominalism or leave it. We have the naïve axiom

$$Stuff(x) \;\&\; madeof(y,x) \;\&\; z(x). \supset z(y)$$

(e.g. : a lump of hard stuff is hard).

which transmits properties from substances to pieces of them. (Care is needed: steel ships float, for example; a fact which often amazes young children.) Notice this axiom is first-order (in a sugared syntax). Quantity is now a function from (pieces)X(stuff) to some scale of measurement, so we can express conservation of quantity through some physical alteration Q by:

$$quantity(piece,stuff) = quantity(Q(piece),stuff).$$

And so on. This works up to a point, but seems to me to be essentially unsatisfactory.

There is a close analogy between being made of a substance, and being made up of a number of parts. And a corresponding analogy between quantity (of stuff) and number (of parts). Sand and piles of small pebbles are intermediate cases: and we often treat an assembly of individuals as a fluid, e.g. as in "traffic flow". The major difference seems to be that different scales of measurement are used in common-sense reasoning (but not in physics, where quantity is number of atoms), as the "paradox of the

heap" shows. This runs as follows: a heap with one stone in it is small. If you add just one stone to a small heap, it's still a small heap. Hence by mathematical induction all heaps are small. The 'paradox' comes by switching from the informal quantity scale of 'small-large' to the precise number scale. Induction is not valid in the former, which (for example) exhibits hysteresis.

Things are often made up of parts joined or related in some way. Obvious examples are physical objects made of pieces glued or assembled together: cups, cars, steam engines, animals. But there are others: processes made up of subprocesses; time-intervals made up of times. The idea of organised collections of entities being regarded themselves as entities permeates our thinking.

Now this fact strikes at the root of an 'individual-based' ontology in the same sort of way that substances do. The only way of handling collections is to count both the collection and its parts as individuals, related by some sort of made of or has-as-part relation. But then these assembled individuals behave in odd ways: they sometimes merge (two heaps make one heap) like pieces of stuff: sometimes they can be disassembled, cease to exist for a time and then perhaps be reassembled: is it the same individual? (Our intuition says: yes, in most cases).

Modal logicians now have very elegant semantic theories which can accommodate such odd behaviour in individuals. But these allow any pattern of vanishing, reappearing and changing properties. The point is to find a way of representing the fact that composite individuals have this special way of vanishing (being taken apart), and to distinguish, for example, those composites which cannot be reassembled (animals, cups) from those that can (cars, steam engines): and to do all this in a framework which assumes that things, by and large, don't just vanish and reappear spontaneously. Composites are thus a different sort of individual, in a very deep sense.

A related issue is how to state criteria upon which we reify a collection into a composite individual. Physical compactness is sometimes sufficient (a heap), but not always necessary (the wiring system of a house), for example. Of course, one does not expect a single general answer, but I do not know of any reasonable answers at all, even for special cases.

I have already remarked on the similarities between being made of (stuff) and being made up of (parts). Is this anything more than a facile analogy? Is there some common framework in which the fundamental ontological notion, rather than existence, is space-occupancy? It might be useful to strive for a representation which allowed the simultaneous expression in different schemes of both 'existence' and 'space-occupancy'. (The schemes would, I believe, have to be essentially different.) Indeed, in a crude way one can see how it might be done directly by "arrays of facts": the array subscripts give one access via spatial relationships to the local presence of objects, which also partake of relationships (represented by a network, say) between themselves and other, non-space-filling, individuals (such as colours). Decomposability is indicated in the array also by 'break lines' which separate the space into regions: different sorts of connection could be fairly easily handled (glued, detachable ...). But this is very crude and has several crucial drawbacks (notably plasticity: imagine moving an object through the space, preserving its shape.)

7. Some non-issues

7.1 Irrelevant classifications

Much heat is generated by disputes based on classifications which do not correspond with the facts, or which at least have outlived their usefulness. Tow such are the "generality vs. expertise" debate and the more recent "procedures vs. assertions" debate. Both of these arise from a revulsion against a particular early naive idea about how to organise intelligent programs, which one could (perhaps unfairly) call the general problem-solver fallacy. (Seymour Papert calls it, the blinding white light theory.)

This was the early insistence that problem-solving methods had to be wrapped up in black boxes called problem-solvers, whose (only) input was a problem and whose (only) output a solution. Problem-solvers were supposed to be as powerful and as general as possible. One had not to "cheat" by "giving" the problem-solver the solution in any sense, e.g. by reprogramming it or cleverly coding the problem in some way (this is made explicit in [7]). Unfortunately, of course, this collection of rules means that there is no way of getting subject-matter-dependent knowledge into the black box; for it cannot be there a priori (violates generality), and it cannot be put into the problem (cheating), and there aren't any other inputs. This is a caricature, but not much of a caricature. Much work in automatic theorem-proving was done with the implicit idea that the theorem-provers were to be regarded as problem-solvers in this sense (c.f. the widely felt 'need' for adequate criteria of relative efficiency of theorem-provers: "my problem-solver is more powerful than yours". (See [2, 10] for a fuller discussion).

The MIT school have now succeeded admirably in destroying this idea, but unfortunately have gotten it confused with some others. Surely we need both generality and expertise: the fallacy is not the emphasis on generality, but the insistence upon the black box and the "no cheating" rules. The general mechanisms of means-end analysis, heuristic search and computational logic should not be rejected, but rather incorporated into more flexible systems, rather than wrapped up in closed 'problem-solving subroutines' or 'methods' or whatever. Thus, to reject conventional uniform theorem-proving systems because they work with assertional rather than 'procedural' languages, is to miss the point. (Whether a language is considered to be a programming language or not, is largely a matter of taste, in any case. LISP can be regarded as (an incomplete) higher-order predicate calculus, or as a first-order applied predicate calculus: predicate calculus can be regarded as a programming language, although by itself not a very good one.) The force of the MIT criticism of computational logic is directed against the 'problem-solver' view and its consequences, especially the lack of any accessible and manipulable (programmable) control stucture in conventional theorem-proving systems. The GOLUX system referred to earlier is an attempt to fill this lack directly with an especially devised control language.

A more recent attack on conventional theorem-proving [7] is that it is too concerned with "machine oriented" logic, and not enough with "human oriented" logic. I confess to being quite unable to understand what this could possibly mean.

7.2 Semantics

Some authors, usually concerned with comprehension of natural language,

use 'semantic' as a vague term roughly synonymous with 'to do with meanings', where this means the same as 'not to do with grammars'. This follows a long and honourable tradition in linguistics (c.f. the use of such terms as "semantic markers" and the idea that linguistic deep structure is semantics).

I wish to emphasise however that this is not the same usage as that adopted here and in formal logic. And it is, I believe, very misleading. It militates against an understanding of the fundamental point that the meanings of linguistic expressions are ultimately to be found in extra-linguistic entities: chairs, people, emotions, fluids.....

As a recent example, Wilks' "semantic units" [24] are syntactic objects in a scheme; nowhere does he tackle the difficult and vital problem of describing exactly what sorts of extra-linguistic entities his "semantic units" refer to. It is easy to say: we must have substances and things and ... ; but what are these? There does seem to be the beginnings of some sort of sketchy semantic theory behind Wilks' formulae (actions have agents which are animate, etc.), but it is not articulated: and if it were, all the problems I have discussed would promptly appear. Similar remarks apply to Schank's work [20], and others.

I am not arguing that natural language should be given an extensional semantics. I distinguish sharply between a natural language, which is an informal and probably not even completely defined means of communication in the real world (is "Eh?" a sentence? Eh?), and a formal deductive scheme for representing knowledge. (It has been suggested to me that the distinction may be related to Sassure's distinction between Langue and Parole, but I have not investigated this.) I suspect that those who deny the usefulness of extensional semantics would also deny the validity of this distinction. That is probably a perfectly respectable philosophical position: but I submit that it is bad engineering.

7.3 Fuzziness and Wooliness

Several authors have recently suggested that more exotic logics, especially 'fuzzy logic', are necessary in order to capture the essentially imprecise nature of human deduction. While agreeing that we have to look beyond first-order logic, I find the usual arguments advanced for the use of fuzzy logic most unconvincing.

Introspection does not suggest to me that intuitive reasonings are essentially imprecise; still less that they are precise in terms of a real-valued truth-value in the unit interval (which is what fuzzy logic would have us accept). Even ignoring introspection, fuzzy logic does not seem very useful, for where do all those numbers come from? (This is McCarthy's point.)

The typical example brought forward to illustrate the need for fuzzy logic concerns the everyday use of such words as 'large', 'small', 'old', 'expensive'. Now it seems to me that, when I say a heap is small, I mean just that. If asked, "Is what you say true?", I will correctly answer "yes", and become impatient with the protagonist. These are precise words but they refer to vague measuring scales. As remarked earlier, for example, the scale 'small-large' exhibits a different topology from the integers or from real intervals: it is more like a tolerance space [27] and it may have hysteresis (an intermediate heap will be considered small if it began as small and grew, and considered large if it began as large and shrank), and it may have gaps in it. The point however is, that we should keep the vagueness of the scale localised into it, rather than

letting it infect the whole inferential system. This enables different 'fuzzy' measuring scales to coexist, which is important. We should investigate what sorts of measurement scales are useful for various purposes.

The most drastic alteration to the actual logic which seems to be needed to handle words like this is to move from a 2-valued to a 3-valued logic, and it is not absolutely clear that even this small step is really necessary.

The view expressed here is different from the one I held some years ago. I have become more respectful, since then, of the unexplored possibilities of predicate logic.

Acknowledgements

Many people have helped me with conversations, suggestions and criticisms. I would like especially to thank John Laski (section 1); Aaron Sloman (sections 2 and 4); Harry Barrow (section 3); Jim Doran (section 4); Bruce Anderson, Carl Hewitt, Johns Rulifson (section 5); Seymour Papert, Gerald Sussman, Bruce Anderson (section 7.1). More generally, I owe much to many conversations with Richard Bornat, Mike Brady, Jim Doran and Bob Kowalski. Alan Bundy, Aaron Sloman and Yorick Wilks made many useful criticisms on an earlier draft.

References

(1) S. Amarel. More on Representations of the Monkey Problem. Internal Report, Carnegie-Mellon University (1966)

(2) D.B. Anderson & P.J. Hayes. The Logician's Folly. DCL Memo 54, Edinburgh University (1972)

(3) R. Balzer. A global View of Automatic Programming. 3rd IJCAI proc. Stanford (1973) (see "Problem Acquisition", paragraphs 3&4)

(4) E. Charniak. Jack & Janet in Search of a Theory of Knowledge. 3rd IJCAI proc., Stanford University (1973)

(5) D. Davidson. Truth and Meaning. Synthese 17 (1967)

(6) M. van Emden & R. Kowalski. The Semantics of Predicate Logic as a Programming Language.

(7) G. Ernst & A. Newell. Some Issues of Representation in a General Problem-Solver. Proc. Spring Joint Comp. Conf. (1967)

(8) A. Guzman. Computer Recognition of Three-Dimensional Objects in a Visual Scene. Report MAC-TR-59, MIT (1968)

(9) P.J. Hayes. A Logic of Actions. Machine Intelligence 6, Edinburgh University Press (1971)

(10) P.J. Hayes. Semantic Trees. Ph.D. thesis, Edinburgh University, (1973)

(11) P.J. Hayes. Computation & Deduction. Proc. MFCS Symposium, Czech. Academy of Sciences, (1973)

(12) P.J. Hayes. Simple and Structural Redundancy in Nondeterministic Computation. Research memorandum, Essex University (1974)

(13) C. Hewitt. PLANNER. MIT AI Memo 258 (1972)

(14) D. Loveland. A Hole in Goal Trees. Proc. 3rd IJCAI, Stanford (1973)

(15) M. Minsky & S. Papert. Progress Report. AI Memo 252, MIT. (1972)

(16) J. Moore & A. Newell. How can Merlin understand? Internal memo, Carnegie-Mellon University (1973)

(17) A. Nevins. A Human Oriented Logic for Automatic Theorem Proving. MIT AI Lab. Memo 268 (1972)

(18) A. Rosenfeld. Isotonic Grammars. Machine Intelligence 6, Edinburgh University Press (1971)

(19) E. Sandewall. Representing Natural Language Information in Predicate Calculus. Machine Intelligence 6, Edinburgh (1971)

(20) R. Schank. The Fourteen Primitive Actions and their Inferences. Stanford AIM-183, Stanford University (1973)

(21) Schank & Colby (eds). Computer Models of thought and language. Freeman (1974)

(22) A. Sloman. Interactions between Philosophy and Artificial Intelligence. Artificial Intelligence 2, (1971)

(23) G. Sussman & D. McDermott. Why Conniving is Better than Planning. MIT AI memo 255A, 1972

(24) Y. Wilks. Understanding Without Proofs. Proc. 3rd IJCAI, Stanford (1973)

(25) T. Winograd. Understanding Natural Language. Edinburgh University Press (1971)

(26) P. Winston. Learning Structural Descriptions from Examples. Ph.D. Thesis, Report MAC-TR-76, MIT (1970)

(27) C. Zeeman. Homology of Tolerance Spaces. Warwick University, 1967

(28) R. Kowalski. Predicate Calculus as a Programming Language. DCL Memo 70, Edinburgh University (1973)

(29) E. Sandewall. The conversion of Predicate-Calculus Axioms, Viewed as Non-Deterministic Programs, to Corresponding Deterministic Programs. Proc. 3rd IJCAI, Stanford (1973)