Int J. Systems Research and Info. *Science, 1985, Vol. 1, pp. 11-22* © *1985* Gordon and Breach, Science Publishers, Inc. and OPA Ltd.
0882-3014/85/0101-0011 $25.00/0 Printed in Great Britain

EXPERT SYSTEMS METHODOLOGY: A CONCEPTUAL ANALYSIS

ROGER T. HARTLEY
Kansas State University, Manhattan, Kansas, U.S.A.

The application of emerging knowledge engineering techniques to expert and consultative systems has tended to out-run the development of a conceptual basis for such applications. This paper presents some conceptual analysis, within the domain of knowledge engineering, of expertise, rule-following and, in particular of competence, which is, perhaps, the crucial concept in the area. Examples are given from MYCIN and CRIB, two diagnostic systems, and Prospector, a consultation system, to support the arguments.

1. INTRODUCTION

1.1 The "accepted wisdom" of knowledge engineering (Davis, 1982a) rests upon several assumptions concerning the nature of expertise within a particular application domain. Although these are often unstated, some are not entirely obvious. They include:

1. The best person to ask about doing a job is an expert.
2. An expert is good at his job, and better than other people.
3. The expert's domain knowledge can be extracted, or elicited and represented on a computer.
4. The expert's heuristic skills can be represented by a set of rules.
5. Facts about the domain can be represented by a data-base of (usually) logical statements.
6. Different experts can come to a consensus of opinion through working with the computer system as moderator.
7. Experts may, over time, come to modify their methods of working to mimic those of the computer.

These seven points are representative of the beliefs and aspirations of workers in knowledge engineering, but I shall not claim that the list is comprehensive. Indeed, in a very young area of computer systems methodology rapid change is inevitable, even at these fundamental levels.

There are many words in the points above that need proper analysis, or at least explanation. The ones I shall discuss are: skill, expert, rule, fact, knowledge. These terms will be analysed as they impinge on knowledge engineering; I shall not here attempt to place them in a wider context, although I believe that this could be accomplished by taking an essentially ethnomethodological approach (see Leiter, 1980, for a good introductory account).

1.2 The notions of expert and expertise will be analysed in terms of *competence*a term which I believe can easily be considered neutral as between human and machine "experts". The relationship between skills and rule-following behaviour will then be examined, leading to the representation of expertise. The Rylean distinction between knowing how and knowing that will be compared to similar distinctions in Al and knowledge engineering. These distinctions and relationships can clearly be seen in existing expert systems.

2. EXPERTS AND EXPERTISE

2.1 The concept of expertise involves two crucial factors: its narrow scope when applied to an individual (or possibly a small group) and excellence in performance. Experts are usually only expert within relatively narrow fields. However, within those fields they not only perform excellently well most of the time, but also much better than the average person. The acknowledged expert is so considered because he is good at his job. Point one above is thus almost tautological. We cannot, of course, ignore the possibility of errors or lapses in judgement; however, experts are *normally* good at their job-they are *competent* in their field. The notion of expertise would not get a hold unless this were so.

2.2 This idea of competence as introduced here seems to me to be the appropriate concept when discussing expertise. It has the right operational connotation for knowledge engineering and avoids the difficulties with the word "intelligent" which has plagued a lot of Al workers. Indeed it is, I think, quite easy to apply the word "competent" to a machine without being forced into unnatural ways of thinking. In this sense competent means "does the job adequately"-a phrase which fits any semi-autonomous machine such as an air-conditioner or lawn-mower just as well as it fits a human-being. A suitably programmed computer can also acquire the attribute by exhibiting its competence in a human field. This is what knowledge engineering is about. In an earlier paper (Hartley, 1982a) I tried to show, following an approach found in Johnson, 1982, that this notion of competence is also an appropriate one when discussing the status of Al programs, vis-ei-vis their structure and performance, when compared with what we know of human-beings in similar situations. Since we are far more interested in what computers do rather than what they are, competence is a far more important idea, as is the associated methodology, *competence modeling*.

2.3 Hand-in-hand with the concept of expertise goes the concept of appropriate domain knowledge. Not only are experts expected to perform better than lay people, but they are also assumed to have more knowledge in the field. This, I would argue, is based on the need to explain why the expert has a better performance; we need to find a basis for their displayed competence over and above the purely mechanical skills which they may possess. For example, consider a person who is an acknowledged expert in legal matters-a lawyer. (Notice that it is possible to define a hierarchy of expertise; our chosen lawyer may not be an expert within the community of lawyers although he is an expert relative to lay people.) Good lawyers possess conversational and rhetorical skills that most people do not. However, we do not consider these to be essential to do the job competently. The necessary skills are those concerning case history and preparation; turning the law to one's advantage; building up experience for future cases. None of these skills are possessed by lay people because they are peculiar to being a lawyer. All, however, involve forms of knowledge which, although they may involve generic properties, have a content which is purely dependent on the law and legal considerations. Without this knowledge the "mechanical" skills of rhetoric are of little use. No judge will accept an argument, however beautifully put, if it is based on false premises. Such a lawyer would surely be branded as incompetent. Thus the more knowledge a lawyer possesses the greater his potential is for competence. I shall turn now to a discussion of forms of knowledge and how these distinctions may be utilised in knowledge engineering.

3. KNOWLEDGE-FORM AND CONTENT

3.1 It is natural in Al to confuse a concept with its representation inside the machine. This comes about, I believe, not through "natural stupidity" (McDermott, 1976) but the deep-rooted assumption that *man is a machine* and its obverse, *machines can be human*. Without this metaphysics there can be no Al. It is natural therefore to push representations or models of cognitive behaviors to their limits. A similar confusion has, I believe, occurred with Ryle's distinction between *knowing how* and *knowing that* in chapter two of *The Concept of Mind* (Ryle, 1949). Here it is natural to assume that the Rylean distinction is the same as the distinction between declarative and procedural forms of representation. For example:

... one should merely note that knowledge *can* be represented procedurally, as knowledge *how* to do something as opposed to knowledge *that* something is the case. (Boden, 1977, page 142.)

The assumption is that a distinction in the forms of knowledge leads to a necessary distinction in their representations; knowing that such-and-such is true corresponds to a fact and therefore to declarative form (e.g. logic); knowing how to do something corresponds to a skill and thus to a procedural form (e.g. a mode of inference or an algorithm).

3.2 The assumption is, however, premature. The procedural/declarative controversy was about representation, not forms of knowledge. The conclusion that knowledge (of whatever form) can be represented using either scheme says nothing about the forms of knowledge involved. It can perhaps best be used as a heuristic spur to better analysis, as are all the best experimental results. The attempt to promote either scheme as better than the other (essentially a reductionist argument) was bound to fail just as a similar attempt to show that LISP is "all data" or "all program" would fail.

3.3 Turning to the Rylean distinction, there have been similar attempts at a reductionist account (Brown, 1973; Hartland-Swann, 1956 amongst others) using philosophical and linguistic arguments. However, all writers acknowledge that the distinction is maintained at the level of everyday speech or for particular purposes such as education (Roland, 1965; Fen, 1966). For example:

If my account is right, all knowing how is knowing that.... At the same time it is evident that Ryle has brought to light important differences between kinds of knowledge, and roughly speaking between two main kinds of knowledge. (Brown, 1973, page 242.)

and also:

It would seem no more desirable to teach mathematical or historical facts as if they were skills like swimming than to teach swimming as if it were Latin or geometry. (Roland, 1965, page 384.)

3.4 In a similar way, the reductionist account does not help the knowledge engineer. His job is not to dictate theory to the expert with whom he is working but to elicit knowledge of whatever sort the expert uses. Since the commonsense view is the same one that Ryle expounded, that knowledge does come in two main sorts, then it is right for the knowledge engineer to work with the same categories. Incidentally, this approach accords with the general line of ethnomethodologists. This emphasis on maintaining commonsense distinctions gives knowledge engineering its empirical flavor. We do not attempt to formalise the problem space and search for analytic solutions as, for instance, in operations research. Instead we take the expert's knowledge more or less as at is (see the next section) and represent it on the computer using whatever techniques are most appropriate. Interestingly, if the expert is found to work completely analytically, then the knowledge engineering approach can, potentially, work just as well as in the more imprecise and informal areas. Knowledge engineering thus overlaps software engineering to an increasing extent, and is coming more and more into the mainstream of computer science.

3.5 The consequence for knowledge engineering of Ryle's distinction is that elicitation of knowledge must look for two sorts of knowledge, especially where the domain of application involves actions, decisions or modes of inference. Elicitation of knowing that will then be done in the form of *facts* about the domain. Elicitation of knowing how will be done in the form of *rules*. It is this process of elicitation which does, I think, give rise to the attempt at a reductionist account. Quite clearly both facts and rules can be formulated as logical propositions (albeit with different forms). Thus:

What my account of practical knowledge shows, however, is that the relevant "knowing how" is still propositional and still formulable in a "that" clause. (Brown, 1973, page 248.)

This reduction is useless for knowledge engineering since there is no mention in these accounts of how the logical propositions are to be interpreted as facts or rules. While the philosopher can assume this interpreter (i.e. the mind of his reader), the knowledge engineer has to *represent* it. Better then to represent facts as static data-structures and rules as programs. It is, of course, possible to choose a uniform representational scheme for both data-structures *and* programs, but this is not necessary, even though it may be desirable from a pragmatic point of view. A good example of this is the production-rule sort of system which can be used to represent either facts or heuristic rules. It is unfortunate that the term "rule" is used in both epistemological and representational settings, but they are, after all, closely related. A heuristic rule is a recipe for action; a production rule, when interpreted correctly, can model this action.

3.6 If I am right about keeping Ryle's distinction as the only useful one for the knowledge engineer, a word must be said about the concept of meta-knowledge as used in Al (for instance, see Brachman, 1979; Davis, 1982b). The popular conception is that representing knowledge about knowledge gives a system an extra dimension of capability which it otherwise would not have. However, on close examination, the extra dimension is not there. Instead we simply have more of the same. The increased capability comes through better organization of knowledge and not through a different sort as is often claimed. There are indeed two sorts of knowledge, but they are knowing that and knowing how, not knowledge and meta-knowledge. We *can* represent knowledge about *rules* and *rule systems* but this is not yet knowledge about knowledge. In fact, any rule system already contains some sort of representation of its own rule base, otherwise it could not operate. Often it is not an explicit representation, but the interpreter which drives the system (sometimes called the "inference engine") contains a procedural representation of the knowledge needed to make the rules work. Making it explicit, which is what the meta-knowledge systems do by using metarules (i.e. rules which make references to other rules) does potentially improve the system's performance, but it does not necessarily introduce another level of knowledge. Put another way, we could say that knowledge is not knowledge unless it *includes* meta-knowledge which refers to its own content. I cannot therefore be said to know something unless I can recall it, reason with it and have my behavior changed because of it. There may be claims to the contrary, but they can usually be analysed in terms of belief or varying degrees of certainty. For knowledge engineering, rather than worrying about adding a new dimension with meta-knowledge, a concentration on analysing and representing the normative aspects of knowing how would, I think, be more profitable. Since understanding the domain is so vital to expertise, trying to grasp how the experts improve their performance through a better-understanding should lead to better expert systems.

4. REPRESENTING EXPERTISE

4.1 We have now seen how experts are competent because they possess increased domain knowledge of the relevant sort. Our lawyer, for instance, not only knows more law and more case history than we lay people do (knowing that), he also is better able to form legal judgements, to prepare and present legal arguments and to learn from experience in the legal setting (knowing how). In knowledge engineering we wish to make the computer as competent as the expert. Point three of the list of assumptions thus comes into play, i.e. the extraction and representation of the expert's domain knowledge.

4.2 This process is best considered in two separate phases:

1. Representing the expert's knowledge to ourselves, through reflective understanding.
2. Representing this understanding to the machine.

The first is what I have called elicitation; the second can simply go under the title representation (this is usually known as "knowledge representation"). I have purposely not given the elicitation phase an obviously empirical flavor for the following reason. We cannot extract knowledge by pure observation; the notion of knowledge "mining" used elsewhere is, I think, a misleading one. We can only elicit knowledge through a process of understanding on our (the knowledge engineer's) part. We inevitably form judgements as to what the expert needs to carry out his job. The expert will also form similar judgements; experts are notoriously bad at formulating their own methods of working-they too need to reflect and understand before being able to speak. Thus through a collaborative dialogue, the knowledge engineer can arrive at an elicitation which, in his view, accurately reflects the expert's knowledge. It will, however, contain a large slice of commonsense, if not out-and-out theoretical input on his part. I shall return to this point later to discuss the normative aspects of elicitation. This will impinge on points six and seven in the assumptions.

4.3 The next phase in system generation is representation of the elicited knowledge to the machine. We have already seen how it is natural to formulate facts and rules during elicitation. This leads on to those representations using data-bases and production systems which cover the bulk of expert systems developed so far. But notice that there is no compulsion to do so. It would be possible to take an elicitation and write a "normal" program. However, it is clear that the use of structured data-bases (semantic-network or frame-based usually) and production systems is far 'More appropriate. It could even be said (as in Hartley, 1982b) that elicitation itself can be guided by the success of these knowledge representation techniques. It is a happy fact that elicitation and representation techniques fit together so nicely. It does not matter which came first-the important thing is that the one supports the other in a classical Al fashion.

4.4 It is possible that the two phases of system generation will run counter to one another. The restrictive nature of the machine and our means of communication with it place stringent demands on the knowledge engineer to keep things as simple as possible. This is of course not necessarily a bad thing since the better systems will be transparent to the user; the structures and procedures within the system had better be easy to understand or the user's cooperation can be lost. Although knowledge representation languages are much improved (Bobrow, 1977; Charniak, 1981; Brachman, 1979) it is still quite a struggle to turn an elicitation into a representation. The newer descriptive languages (e.g. DL in Winograd, 1982; and Hartley, 1983) hold out hope however for making this job easier, and computer-aided acquisition of knowledge is an added possibility (Davis, 1982b). But without such help the best that can be done is to guide elicitation by consideration of the machine's characteristics and of available representation techniques. It is possibly this pragmatic solution which has led to points four and five in the list of assumptions. The argument (unstated) goes like this: knowledge representation is hard; the machine can (we know) support representations of rule systems (i.e. programs) and of logical statements (i.e. data-bases); therefore if we can elicit knowledge in corresponding forms, then representation is made easier. The proliferation of rule-based systems and semantic-network or frame-like databases can thus be seen as a practical solution to a difficult set of problems.

5. EXPERTISE AND RULE-FOLLOWING

5.1 The success of expert systems using the rule-based approach prompts the question as to whether experts themselves follow rules just as the computer follows the program. If this were so the pragmatic solution to the representation problem would have a very nice theoretical underpinning. It would also add fuel to one of the metaphysical debates within Al-whether man is a machine. To discuss this possibility further we need to draw distinctions between three sorts of rulefollowing behavior. Type one (RFI) is the most obvious sort-following an explicitly stated, external system of rules. Examples of this simple sort of behavior are solving some kinds of symbolic problems (calculus and logic to name two); checking out the systems of a 747 aircraft before take-off, processing a social security application. The rule system has been arrived at differently in all three cases, but each system is finite and considered to be heuristically adequate in each case. The rules of a formal system are arrived at by consideration of what Rescher calls cognitive systematization. They are simple, regular, complete, coherent and effective-all attributes of a "good" system (Rescher, 1979, pages 10 and 11). The 747 take-off procedure rule book is derived from considerations of functionalityall sub-systems must be considered working before take-off can go ahead. The social security application is processed according to legal, administrative and social rules-a very complex mixture indeed.

5.2. However, all of these jobs require a level of expertise above that of the average person, but yet can be executed by rigorously following the "rule-book". The expertise comes not so much in problem-solving ability (although this may be needed when the rule book fails) but in "working" the rule book. Experience counts for a lot here.

5.3 Type two (RF2) behavior is very similar to RFI but here the rule book is "internalised" and does not exist at all in external form. This form of behavior covers most of the areas of expertise considered to be appropriate for knowledge engineering applications. Diagnosis, whether of faulty bodies as in medicine, or faulty machines, as in engineering, is the prime example of RF2. A successful (i.e. competent) diagnostician operates systematically without being algorithmic and hence inflexible. The knowledge he uses is, however, not innate; usually a long training period is necessary to acquire sufficient competence. Learning must occur within a process of systematization and it is natural to attempt to "lay down the law" to make the systematization more effective. Thus the effort in making sense of a complex semi-analytic domain naturally involves the learning of a set of heuristics or rules-of-thumb without which the job would be impossible. Knowledge engineers are in the process of making some of these heuristics explicit, but most are handed down by word of mouth; having been learned through experience they are passed on as rules not to be obeyed blindly, but to serve as an aid and guide. Perhaps a good analogy to explain the difference between RFI and RF2 is between deterministic algorithms on the one hand (RFI) and non-deterministic algorithms on the other (RF2). They can both be programs (rule-systems) but their behaviors are different, as are the methodologies behind them.

5.4 The third type of rule-following (RF3) is far more controversial in that it attempts to reduce all behavior to rule-following. The prime example of this is Chomsky's theory of innate knowledge of a language (see Chomsky, 1967, for a good account of this). He attempts to make a case for the learning of a language (when a child) as governed by trial-and-error applications of the rules of the language (its grammar) which are assumed to be "known" already. Thus when we speak we are following the rules of the language, present only in our subconscious mind. This theory has been extended to cover other aspects of knowing how; language production is a skill and learning the skill is acquiring the knowledge of how it should be done. RF3 can thus be seen as a general theory of knowing how; the extension can be made by noting the undeniable similarities between language production and more obviously learned skills (like typing or solving crossword puzzles) i.e. the role of memory, trial-and-error learning and reinforcement through repetition. Returning to expertise, RF3 could be presented as a theory to cover those areas where no explicit "rule book" is apparent. We might then be tempted to say that elication of knowledge is really *uncovering* the system of rules that the expert uses, even when he is unaware of its existence. However, I believe that this is a mistaken view. RF3 is a discredited theory (for instance, see Cooper's attack on Chomsky in Cooper, 1975) since the notion of innate knowledge cannot be supported fully when faced with a barrage of philosophical and psychological evidence to the contrary. Moreover, we do not need to claim that knowledge engineering is aimed at modeling RF3, as we would if the theory were to be upheld. A sounder approach is to say that we aim to represent knowledge as if *it were* RFI. The resulting system does not model the expert's knowledge directly, but does so only indirectly through our reflective understanding of his competence and the possible reasons for it. We therefore attempt to generate a rule set which, if it were to be followed like a rule book, would reproduce the important parts of the expert's behavior.

5.5 A knowledge engineering system is thus best viewed as a *competence* model. It is not a model of a particular expertise as possessed by an individual, but it presents a normative account of competence in the chosen area. The use of an expert or group of experts in the elicitation phase does not necessarily imply otherwise. In fact elicitation from experts is the easiest and quickest way to first formulate, and then validate our theoretical ideas. The rule system so developed need not *necessarily* represent any "rule book" possessed by the expert, be it of type RF1, RF2 or even of RF3. However, it does provide a degree of commonality amongst experts (point six in the assumptions) where methods may differ even though results agree. It also gives the possibility (point seven) of better performance than that of the experts from whom the initial elicitation was done. This could not be the case were we to restrict ourselves to modeling RF3 behavior.

5.6 The mere possibility of attaining a state in which assumptions six and seven might come into play is not to be underestimated. It is not too wild to speculate that computer programs such as those described here can be used as a focus for both academic and practical progress in the fields in which they operate. The computer then takes on a new role, not as number cruncher or as data bank, but as a *document* i.e. as a dynamic vehicle for new ideas and the models which make the ideas explicit. It could be argued that this is exactly what Al has been doing for twenty-five years in the field of cognition. The utility of Al programs lies not in their performance (which was often patchy, if not downright threadbare) but in their uncanny potential for provoking thought leading eventually to better theories. The knowledge engineering program can achieve a similar effect by making normative statements in its field. An explicit competence model can help even an expert become more competent.

6. **THE METHODOLOGY AT WORK-THREE EXAMPLES**

6.1 This final section will attempt to place expert system methodology in a more down-to-earth setting. Three systems will be examined with a view to explaining their form and content in the light of the preceding discussion. All three are *consultation* systems which work *with* experts rather than attempting to replace them. **MYCIN and CRIB** are diagnostic systems, in the areas of faulty bodies and faulty computers respectively. Prospector is an advice-giver in the area of mineral exploration. There is of course a danger that I am interpreting history to fit ideas only developed later. In other words I may say that the designers did such-and- whereas, in fact, they did not. However, the aim is not to reinterpret history, but to provide latter-day justification for our earlier efforts, which were often illformed and evolutionary in character.

6.2 Firstly, what can we say about each of the tasks independently of the models presented by each system? All three have the same characteristics regarding their necessary input and output data. The input to all of them is a result of real-world observations. These observations may have stemmed from physical monitoring and measuring devices (patient's blood-count, local voltage, seismological data) but they can all be represented propositionally in the form of "known" facts. The output can also be characterized as similar for each task. The end result is a piece of advice of the form "it is my best judgement that you ought to . . ." where the advice may be "take this drug", "change this board" or "dig there". The differences only emerge when looking at the details of how the conclusion is arrived at, what the form and extent of interaction with the world is, and who the conclusions are aimed at. Diagnosis, for instance, may seem to be a task that can be easily characterized by an information-processing model. Crudely put, this says that the expert operates as a black box whose inner workings are irrelevant to the production of the final piece of advice. However, a doctor who operates exclusively in this manner is often branded as incompetent simply because the competence of a doctor is not only measured by the efficacy of his remedies, but instead by his manner, his evident thoroughness, his frankness and his compassion. However, technically, the sort of diagnosis modeled in **MYCIN and CRIB** is essentially the same. Both operate using hypotheses which are confirmed or disconfirmed by inference supported by further, well directed measurements. Doctors and computer engineers often use a "hunch" mode because experience tells them that searching for possibilities is considerably shortened that way. They also rely on diagnostic methods which can be memorized (when is the last time you saw a doctor use a reference book to help him in diagnosis?).

6.3 The case of analyzing geological data for mineral exploration is different in that a battery of complex and expensive tests is usually done in a standard way thus yielding large amounts of information, most of which is only interpretable by computer. The geologist's job is therefore done in isolation, working with purely symbolic information. His knowledge and experience has then to be rather more sure, since he usually cannot use the test/observe cycle of diagnosis. He will often use reference material for geological facts and his analysis may be more exhaustive than is the case **with diagnosis.**

6.4 All three jobs are carried out by highly-trained people. The doctor may have an edge in responsibility, having patient's well-being in their hands, but the level of complexity of the systems with which they work are comparable. All three involve handling uncertain information which needs experience to interpret. The type of uncertainty is different in each case, but the way in which it is usually handled is not. Doctors are working with uncertain knowledge about physical and chemical mechanisms; computer engineers handle the uncertainty of interactions among a myriad of simple devices whose characteristics are well-understood; the geologist simply cannot see what is underground and must work with second-hand and partial information. All these uncertainties are however attacked with general methods of analysis according to laws and models, hypothesis formation and testing and case history experience. It is these methods which are embodied in the programs described below.

6.5 MYCIN (Shortliffe, 1976) is probably (and deservedly) the best-known of all expert systems. Its architecture is well known as an example of a backwardchaining production system. For our present purposes though, the program presents a model of diagnostic competence with the following set of features:

A. Elicitation was done using a community of experts (doctors) to yield a set of facts to model judgemental knowledge (knowing that), each with an attached certainty factor, and a strategy for diagnosis using these facts (knowing how).

B. Uncertain facts are represented procedurally as rules of the form: IF premises THEN conclusion with certainty factor X. Certain facts merely have the maximum value of the certainty factor. MYCIN thus presents a model of experience with diagnostic information; there is no hint that the doctor's belief system is being modeled.

C. The diagnostic strategy is relatively fixed and takes the form of hypothesis confirmation or disconfirmation. This strategy (implemented by backward chaining through the rules) is represented partially through "meta-rules" but mainly in the code of the production system interpreter. Meta-rules of the form: IF rule base contains such-and-such **THEN alter future program** control flow appropriately, can change the tactics of **proving a partial premise** (in order to confirm a conclusion) but not the overall strategy. Certainty factors are combined in a commonsense way so that certainty increases for confirmation and decreases for disconfirmation.

D. The competence of the system has been validated by experts (doctors again) and its performance compares favorably with their own.

6.6 Prospector (Duda, 1979) also employs a backward-chaining hypothesis method like MYCIN, but the knowledge base is a partitioned semantic network of connected uncertain facts, not a production system. The features of its competence model are:

A. Elicitation was carried out with the help of geologists to yield a set of facts, each of which, like the MYCIN rules, has an attached certainty factor. The same strategy of hypothesis testing was also employed. Each hypothesis is assumed to denote an instance of a generic geological situation, of which Prospector has a small fixed number. These situations (models of mineral deposits) each describe a subset of the semantic network.

B. Facts are represented as nodes in the network; the arcs connecting them represent possible inferences passing from one fact to another. This architecture is equivalent to the production-rule system in MYCIN. Since at any point in a consultation session, each node has an attached certainty factor, the whole represents the geologist's thinking at that time. Some nodes will have a minimum representing falsehood, some will have a maximum representing truth, and some will be in between.

C. The geologist's method of working is represented as a program-which matches input information against the network to generate a hypothesis (the deposit model corresponding to the best match) and then tries to confirm the hypothesis by backward chaining within the model from known facts to unknown ones. Again the strategy is a fixed one-there are no meta-rules to provide local improvements as in MYCIN.

D. Prospector is currently undergoing enlargement to make it more useful, but limited success has been achieved in real consultations.

6.7 CRIB (Addis, 1979; Hartley, 1984) is a diagnostic system like MYCIN, but attempts to mend computers, not cure diseases. The knowledge base is a model of a computer engineer's experience in locating faults within physical sub-systems of the machine given a number of observable symptoms. The model of competence it presents includes these features:

A. Instead of using experts exclusively, elicitation was initially carried out from an intermediate form: a set of fault-finding guides (type RF1 knowledge). These guides *prescribed* a diagnostic strategy (essentially an informal decision tree) and assumed that relevant facts about the machine were known. The elicitation from these guides (and from experts as well) took the form of production rules which paired sets of symptoms with faulty sub-systems.

B. Facts in CRIB are assumed to be of two sorts. Those symptom/faulty system pairings which lead to a successful diagnosis are considered to be certain. They model the represented by pairings where the symptoms form a subset of those in a certain pairing. They form the basis of a heuristic strategy; they can be considered as uncertain facts where the uncertainty is represented by the number of times the pairing has been used successfully.

C. CRIB represents the diagnostic strategy as a program, as does MYCIN, but interestingly, the search/match procedures are executed in hardware on the CAFS device, leaving the programmer free to model the heuristic aspects of the strategy. A complete match with a symptom/sub-system pair is sought, but a partial match is more likely. CRIB then employs a set of heuristics for inviting the engineer to test for the missing symptoms in order to complete the match and thus prove the fault. This search is guided by attempting to maintain a downward progression down a hierarchical model of the computer's hardware systems and their interconnections.

D. CRIB's competence has been tested by engineers and it has performed satisfactorily within a limited range of known faults. Like both MYCIN and Prospector, though, it will flounder when presented with incorrect information, or information leading to conclusions unknown to the system.

6.8 The above brief descriptions are unexciting for anyone who is familiar with any of the systems mentioned and probably insufficiently detailed for everyone else. However, it is more the form of the descriptions which is important in this paper, not their content. Each of the three systems operates in a completely different area of expertise, yet their representational schemes, their elicitation methods and results and the recognition of their competence by the relevant experts are all remarkably similar. As much could not be said about the similarities between programs from, say, the fields of commerce. Knowledge engineering is already building a well-defined methodology which is applicable to widely-differing areas. It is possible, however, that "normal" commercial programming could also benefit from the sort of methodology outlined here. Certainly such systems could be labelled competent (or incompetent) by the same sense as I have used the term in connection with expert systems. Perhaps many "straightforward" systems can also be written as applications of knowledge engineering. That, although I believe it strongly, is the subject of another paper.

7. CONCLUSION

7.1 This paper has attempted to find a basis for the continued application of the accepted wisdom of expert systems methodology. This is just as important for knowledge engineering as it is for any sort of engineering. The analysis is not necessarily a prescription for action, although, because Al is a field where studying human behavior is considered by many to be a prerequisite for a good system design, it is not out of order to study the ways in which behavior is studied. The sorts of philosophical references used are thus to be regarded as having many relevant things to offer at the conceptual level even if they say nothing at more practical, down-to-earth levels.

7.2 The picture here presented of expert systems is of models of competence, not of simulations and not analytical models in the formal system sense. This conclusion alone puts knowledge engineering and expert systems methodology, its means of application, into a new category of computer use. It is hoped that conceptual thinking will keep up with our practical activities to turn the whole package into a solid contribution to science and technology. It certainly deserves to be so considered.

REFERENCES

Addis, T. R. and Hartley, R. T. (1979), A fault-finding aid using a content-addressable file store. TN 79/3 International Computers Limited, Stevenage, U.K.

Bobrow, D. G. and Winograd, T. (1977), An overview of KRL, a Knowledge Representation Language, *Cognitive Science 1(1)*

Boden, M. (1977), *Artificial Intelligence and Natural Man*, Harvester Press, Brighton.

Brachman, R. J. (1979), On the epistemological status of semantic networks, in: Findler, N. V. (ed.), *Associative Networks: Representation and Use of Knowledge by Computers,* New York: Academic Press.

Brown, D. G. (1973), Knowing how and knowing that, what, in: Ward, 0. P. and Pitcher, G. (eds.), *Modern Studies in Philosophy: Ryle.*

Charniak, E. (1981), A common representation for problem-solving and language comprehension information, *Artificial Intelligence* 16(3).

Chomsky, N. (1967), Recent contributions to the theory of innate ideas, *Synthese 17(l).*

Cooper, D. E. (1975), *Knowledge of Language,* London: Prism Press.

Davis, R. and Buchanan, B. B. (1977), Meta-level knowledge: Overview and applications, *Proceedings of IJCAI 5*. Cambridge, MA, pp. 920-927.

Davis, R. (1982a), Expert systems: Where are we? And where do we go from here?, *AI Magazine* 3(2).

Davis, R. (1982b), Teiresias: Application of meta-level knowledge, in: Davis, R. and Lenat, D. B., *Knowledge-based Systems in Artificial Intelligence,* New York: McGraw-Hill.

Duda, R., Gaschnig, J. and Hart, P. (1979), Model design in the **PROSPECTOR** Consultant System for mineral exploration, in: Michie D. (ed.), *Expert Systems in the Micro-electronic Age*, Edinburgh: Edinburgh University Press.

Fen, Sing-Nan (1966), "Knowing that" rediscovered and its place in pedagogy reassigned, *Educational Theory 16*.

Hartland-Swann, J. (1956), The logical status of "Knowing that", *Analysis 16*.

Hartley, R. T. (1981), How expert should an expert system be?, *Proceedings of IJCAI 7,* University of British Columbia, Vancouver, Canada, 862-867.

Hartley, R. T. (1982a), The competent computer. MCSG/6 Man-Computer Studies Group, Brunel University.

Hartley, R. T. (1982b), A conceptual basis for expert systems methodology, *Proceedings of EXPERT SYSTEMS 82,* ACM-IEE-SPL, Brunel University, September.

Hartley, R. T. (1983), A virtual machine design for a description language, Technical Report, Computer Science Dept., Kansas State University.

Hartley, R. T. (1984), CRIB: Computer diagnosis through knowledge engineering, *IEEE Computer,* March, 1984, 76-83.

Johnson, L. (1982), Practical reasoning, in: Trappl, R., Ricciard, L. and Pask, A. (eds.), *Progress in Cybernetics and Systems Research*, *Vol. IX* New York: Hemisphere, 351-356.

Leiter, K. (1980), *Primer on Ethnomethodology,* Oxford: Oxford University Press.

McDermott, D. (1976), Artificial intelligence meets natural stupidity, *SIGART Newsletter 57*, April, 1976. (Reprinted in *Mind Design* (Haugeland, L., ed.), Cambridge: MIT Press, 1981).

Rescher, N. (1979), *Cognitive Systematization,* Oxford: Blackwell.

Roland, J. (1965), On "Knowing how" and "Knowing that", *Philosophical Review 67.*