A CONCEPTUAL BASIS FOR

EXPERT SYSTEMS METHODOLOGY

Roger T. Hartley
Kansas State University

Conference Topic: Applications Systems

Abstract

The application of 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 the computer fault-diagnostic system, CRIB, to support the argument.

1.0 Introduction

1.1 The 'conventional wisdom' of knowledge engineering [11 makes several assumptions concerning the application domain and the nature of expertise in the domain. Some are obvious, some not so. 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 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 agree through working with the computer system.
7. Experts may, over time, come to modify their methods of working to suit that 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 exhaustive. Indeed, in a very young area of computer systems methodology, rapid change is inevitable, even at this fundamental level.

1.2 Point 1 is almost tautological; perhaps it is best considered as a definition of the word expert. But point 2 is more important when considering expertise in general. The acknowledged expert is so considered because he is good at his job. We cannot, of course, ignore the possibility of errors in judgement, but we can state with certainty that experts are normally good at their job - they are competent in their field. The notion of expertise would not get a hold at all unless this were so.

1.3 Hand-in-hand with the concept of expertise goes the concept of appropriate domain knowledge. Experts are expected and assumed to have more knowledge than other non-experts. This, I would argue, is because we need to find a basis for their displayed competence over and above the purely mechanical skills which experts also possess. For example, consider any lawyer, an acknowledged expert in all matters legal. (Note that expertise is also a relative thing - he may be expert to a lay-person, but an also-ran within the community of lawyers. He cannot, however, be incompetent.) Good lawyers possess conversational and rhetorical skills that most people do not, but we do not consider these

to be necessary to the job. The necessary skills are those concerning case research and preparation, turning the law to their advantage,

building up reliable experience for future cases. All of these concern knowledge in one form or other and it is really these forms of knowledge which concern me most in this paper.

2.0 Representing Expertise

2.1 Thus, experts are competent because they possess increased domain knowledge. By knowledge here I mean both forms as discussed by Ryle [31]. The distinction between knowing That and knowing how is an excellent one to talk about knowledge engineering. Our lawyer in the example above must not only know case history and the law, but also be able to form judgements, prepare (and present) arguments, and learn from experience. The former is knowing that, the latter is knowing how.

2.2 In knowledge engineering we wish to make the computer as competent as the expert. ([4] presents a case for the 'competent computer'.) The major aim is therefore point 3 of the list in the production; the extraction and representation of the expert's domain knowledge. If the computer is to be considered competent, we must try to capture as much as possible of the expert's knowledge. Thus, knowledge engineering has a distinct empirical flavor to it. We do not attempt to formalize the problem space and search for analytic solutions as, for instance, in operations research (the so-called power-based approach). Instead, we take the expert's knowledge more or less as it is (I have more to say on this in the next section) and represent it on the computer, complete with ambiguities and inconsistencies. 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 difficult areas. Two problems then face us when engaging in this process of representation:

1. Representing the expert's knowledge to ourselves (through reflective understanding).

2. Representing this understanding to the machine.

This breakdown of the knowledge engineer's job, into elicitation followed by representation has been formulated many times before [eg., 9, 10]. However, I think that 1 and 2 above are a more accurate reflection on what is involved than the notion of knowledge 'mining' used elsewhere. We cannot 'extract' knowledge without understanding on our part. We inevitably form judgements as to what the expert needs to carry out his job; we cannot be said, therefore, to be representing the expert's knowledge on the machine, but rather our understanding of it. It is of equally no use getting the expert himself to engage in the process of elicitation. It is acknowledged that experts normally find it hard to express their knowledge in any other way than through their competence. Even they have to go through the two steps outlined above.

2.3 The two steps presented above almost always run counter to each other due to the restrictive nature of the machine and the means of communicating with it. Although knowledge representation languages are much improved [6, 7, 8], it is still a struggle to turn an elicitation into a representation. The solution to this problem is obvious: guide the elicitation by consideration of the machine's characteristics. It is this solution which, I believe, has led to points 4 and 5 in the conventional wisdom. The argument 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 expert knowledge in these forms, then knowledge representation is made easier. The proliferation of rule-based systems and semantic-network or frame-like data bases is thus seen as a practical solution to a difficult set of problems.

3.0 Expertise And Rule Following

3.1 The success of expert systems using the rule-based approach prompts the question of whether experts themselves follow rules just as the computer follows the program. To discuss this point we need to draw a distinction between three different sorts of rule-following behavior. Type 1 (RF1) is the most obvious sort; following an explicitly

stated, external system of rules. Examples here are solving some kinds of symbolic problems (calculus, logic); checking out the systems of a 747 before take off; processing a social security application. 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". Type 2 (RF2) is very similar to RFD but here the "rule book" is internalized and does not exist in an explicit 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. An expert in diagnosis (and other similar semi-analytic domains) operates systematically without being algorithmic and, hence, inflexible. The best analogy is between RF1 and deterministic algorithms on the one hand and RF2 and non-deterministic algorithms on the other.

3.2 The third type of rule following (RF3) is far more controversial in that it attempts to reduce AU behavior to rule-following. The prime example of this is Chomsky's theory of innate knowledge of a language [111. 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. This theory could be extended to cover other aspects of knowing how, since language production clearly requires some such knowledge. RF3 can thus be seen as a general theory of knowledge how. Returning to expertise, RF3 could be presented as theory to cover those areas where no explicit "rule book" is apparent. We might then be tempted to say that elicitation 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 (see Cooper's attack on Chomsky in [12]) since the notion of innate knowledge cannot be supported fully. Moreover, we do not need to claim that knowledge engineering is aimed at modelling RF3. A sounder approach is to say that we aim to represent knowledge as jS it was RF1. The resulting system does not then model the expert's knowledge directly, but only indirectly through our reflective understanding of his competence and the possible reasons for it.

3.3 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 6 in the conventional wisdom) where methods may differ even though results agree. It also gives the possibility (point 7) of performing better (more competently) than the experts from whom the initial elicitation was done. This could certainly not be the case were we to restrict ourselves to modelling RF3 behavior.

4.0 An Example of Expert Systems Methodology

4.1 CRIB [2, 5] was an application of knowledge engineering to the difficult task of diagnosing and mending faulty computers. When we examine the status of the end-product we can now see it as a good example of the methodology discussed above. Information for the system design was gathered from three sources:

A. Fault-finding guides in the form of algorithmic flow charts. This is RF1 knowledge.

B. Interviews with expert computer engineers. It became clear during this period that good engineers were incapable of expressing their RF2 knowledge, even though they clearly worked in a systematic fashion (the same faults found in the same way).

C. Analysisof protocols tape-recorded during actual fault investigations. This provided a different angle on type RF2 behavior.

4.2 The results of putting together information from sources A, B, and C was that, although important as a "pump-primer", the information was useless without a good theoretical understanding of the problem area. This is not to say that an analytic solution should be sought, but merely that there should be some criteria by which to judge the worth of the information collected. Having good understanding of the sorts of knowledge possessed by engineers and how they used it enabled us to build a system which did not reflect totally on their working methods, but used factors induced from the information sources listed above, and pieces of theoretical insight added by us. For instance, in the initial stages of a fault investigation good engineers are guided, not by possible faults, but by actual symptoms. We thus made CRIB entirely symptom-driven, it contains no knowledge of faults at all. We do not claim that engineers do not consider possible faults, but CRIB's competence demonstrates that it is not necessary to do so. It may not even be desirable - this is the normative potential of knowledge engineering again.

4.3 In summary, CRIB is not a simulation model of one or more engineers. It is also not an analytic model of computer diagnosis. The model it does present is one of diagnostic competence. It is a fictional portrait of a good engineer which can also demonstrate its abilities by finding real faults in real situations. Only a system prepared along knowledge engineering lines can claim to do this.

5.0 Conclusion

This paper has attempted to find a basis for the continued application of the conventional wisdom of expert systems methodology. This is just as important for knowledge engineering as it is for any sort of engineering. Too often in the past of computer science and artificial intelligence theory has come along too little and too late, whilst mistakes are perpetrated in the name of expediency. While I feel that this will not happen in knowledge engineering, it is still right, I believe, to explore the concepts involved in order to provide sound support for our practical activities.

REFERENCES

[1] Davis, R., Expert Systems: Where Are We and Where Do We Go From Here? Invited talk, IJCAI-81. University of British Columbia, Vancouver. 1981.

[2] Addis, T. R. and R. T. Hartley, A Fault Finding Aid Using a Content Addressable File Store. ICL Research and Advanced Development Centre, Stevenage, U. K. TN 79/3, 1979.

[3] Ryle, G., The Concept of Mind. London, 1949.

[4] Hartley, R. T., Competence Modelling as a Methodology for Computer Systems. (Communications of the ACM, forthcoming).

[5] Hartley, R. T., How Expert Should an Expert System Be? Proceedings IJCAI-81, University of British Columbia, Vancouver, 1981 , p. 862-867.

[6] Bobrow, D. G. and T. Winograd, An Overview of KRL, a Knowledge Representation Language. Cognitive Science 1(1), IGN, P. 3-46.

[7] Brachman, R. J., On the Epistemological Status of Semantic Networks in Associative Networks: Representatio and Use 91 Knowledge by Computers, N. V. Findler (Ed.) Academic Press, New York. 1979, P. 3-50.

[9] Feigenbaum, F. A., Themes and Case Studies of Knowledge Engineering, In Expert Systems In the Microelectronic Age Michie D. (Ed.), Edinburgh University Press, Edinburgh, 1979, P. 3-25.

[10] Davis, R., Knowledge Acquisition in Rule-Based Systems, in Pattern-Directed Inference Systems. Waterman, D. A. and Hayes-Both, F. (Eds.) Academic Press, New York, 1978, P. 99-134.

[11] Chomsky, N., Recent Contributions to the Theory of Innate Ideas. Synthese 17(1), 1967. p. 2-11.

[12] Cooper, D. E., Knowledge of Language, Prism Press, London, 1975.