Int. J. Man–Machine Studies (1987) **26**, 231–243

Generic tasks for knowledge-based reasoning: the "right" level of abstraction for knowledge acquisition

TOM BYLANDER AND B. CHANDRASEKARAN

Laboratory for Artificial Intelligence Research, Department of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210, U.S.A.

Our research strategy has been to identify *generic tasks*—basic combinations of knowledge structures and inference strategies that are powerful for solving certain kinds of problems. Our strategy is best understood by considering the "interaction problem", that representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and by the inference strategy to be applied to the knowledge. The interaction problem implies that different knowledge-acquisition methodologies will be required for different kinds of reasoning, e.g. a different knowledge-acquisition methodology for each generic task. We illustrate this using the generic task of hierarchical classification. Our proposal and the interaction problem call into question many generally held beliefs about expert systems such as the belief that the knowledge base should be separated from the inference engine.

Introduction

Knowledge acquisition is the process that extracts knowledge from a source (e.g. a domain expert or textbook) and incorporates it into a knowledge-based system that solves some problem. Whether the knowledge is acquired by a knowledge engineer or by a program, ultimately the knowledge must be encoded in some knowledge-base representation. Consequently, knowledge acquisition cannot be separated from a broader theory of knowledge-based reasoning; a solution to knowledge acquisition must be compatible with a solution to the general problem of knowledge-based reasoning.

For some time now, we have been developing a *theory of generic tasks* that identifies several types of reasoning that knowledge-based systems perform and provides a overall framework for the design and implementation of such systems (Chandrasekaran 1983, 1984, 1986). In this paper, we present our theory as a way to exploit the "interaction problem". Because each generic task exploits it differently, each one should be associated with a different knowledge-acquisition methodology.

First, we pose and discuss the interaction problem. Next, we review our theory of generic tasks: the characteristics of a generic task and the generic tasks that have been identified so far. In view of the interaction problem, we propose our theory of generic tasks as a framework for identifying different knowledge-acquisition methodologies. We illustrate this using the generic task of hierarchical classification. Finally, we reflect on a number of beliefs that have driven much of the past research on knowledge acquisition and knowledge-based reasoning.

0020-7373/87/020231 + 13$03.00/0

The interaction problem

The interaction problem is this:

> Representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and by the inference strategy to be applied to the knowledge.

In other words, how knowledge is represented has a close relationship to how knowledge is used to solve problems; knowledge is dependent on its use. The interaction problem is not a new notion. Minsky, in his famous frame proposal, argues that "factual and procedural contents must be more intimately connected to explain the apparent power and speed of mental activities" (Minsky, 1975, p. 211). Marr has noted that "how information is represented can greatly affect how easy it is to do different things with it" (Marr 1982, p. 21). Our argument takes a different perspective, that the problem and the inference strategy influence what knowledge is represented and how knowledge is encoded, i.e. knowledge will be represented to take advantage of what it will be used for.

The interaction problem, if true, has serious implications for how knowledge acquisition should be done. Because some knowledge representation must be the target of knowledge acquisition, knowledge-acquisition methodologies must take the interaction problem into account. Also, if different kinds of reasoning have different kinds of interactions, there is a need for a different knowledge-acquisition methodology for each kind of reasoning.

REASONS FOR THE INTERACTION PROBLEM

Figure 1 illustrates the situation that gives rise to the interaction problem. Starting with a problem and a domain expert (or some other source of knowledge), the goal of knowledge acquisition is to construct a knowledge-based system that solves the problem by the combination of a knowledge base and an inference strategy (a

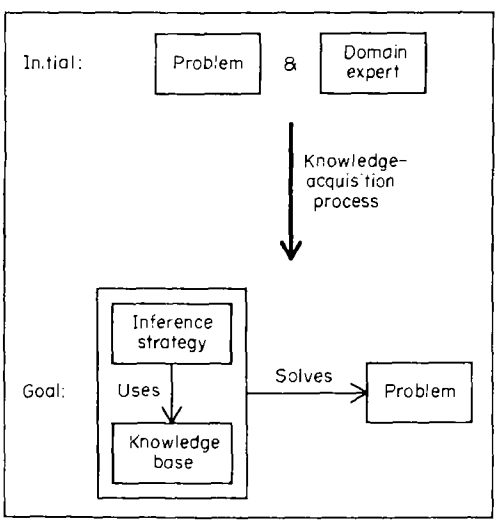

FIG. 1. Initial and goal states of the knowledge-acquisition process.

process to use or interpret the knowledge). There are two primary reasons why this leads to the interaction problem.

(1) Choice of knowledge. The knowledge acquisition process must *choose* what knowledge to ask for and what knowledge to encode. The choice is driven by the need to gain leverage on the problem by obtaining knowledge with high utility and to reduce complexity by avoiding or discarding knowledge with low utility. Not everything the domain expert knows has the same level of usefulness, and in any case, it is not feasible to acquire everything that the domain expert knows.

(2) Constraints of inference strategy. A knowledge representation requires some process that, given a description of a situation, can *use* (or *interpret*) the knowledge to make conclusions. It is this process which we call the "inference strategy" ("inference engine" is an equivalent phrase). The knowledge must be represented so that the inference strategy reaches appropriate conclusions (appropriate to the problem being solved) in a timely fashion. Consequently, the knowledge must be adapted to the inference strategy to ensure that certain inferences are made from the knowledge and not others. Also, give a choice of inference strategies, there will be an interaction between the strategy chosen and the form of knowledge.

EXAMPLES OF THE INTERACTION PROBLEM IN RULE-BASED REPRESENTATIONS

The idea of rules is to explicitly map situations to actions. Naturally then, the focus is on determining what conditions characterize the situations and what conclusions characterizes the actions. The result is that two different problems in the same domain can have different rules representing the "same" knowledge. For example in diagnosis, rules of the form "symptom→malfunction" will be implemented, while in prediction of symptoms, the rules will be in the form "malfunction→symptom". In each case, the knowledge will be adapted to the problem. One might argue that there is no difficulty with keeping both problems in mind, and so both kinds of rules can be in the same knowledge base. Of course, *given that one has already taken the interaction problem into account,* the knowledge base then will have rules appropriate for the problems to be solved.†

Another source of interaction is that special programming techniques are needed to encode problem-specific inference strategies. For example, R1 (McDermott, 1982), which is implemented in OPS5 (Forgy, 1981), performs a sequence of "design subtasks", each of which is implemented as a set of production rules. However, OPS5 has no construct equivalent to a subtask, so the grouping of rules and the sequencing from one set of rules to another are achieved by programming techniques. Clearly, the constraints of OPS5's production rule representation has had a significant effect on how R1's knowledge was encoded.

Different inference strategies for rules are also a source of interaction. If

† It is actually dangerous to have both kinds of rules in a knowledge base. Given some confidence in a malfunction, then some confidence in the symptoms it causes should be inferred. However, one shouldn't infer confidence in other malfunctions that cause the same symptoms.

EMYCIN's backward-chaining strategy is used, rules can combine with other rules to increase or decrease confidence in a given conclusion (van Melle, 1979). On the other hand, if OPS5's recognize-and-act strategy is used, only one rule at a time can be fired, so that situations must be matched to actions much more exactly. Also, the "context" must be carefully controlled to ensure that appropriate rules are considered. Note that the difference is not whether EMYCIN does forward- or backward-chaining, but that EMYCIN allows rules to act in parallel, while OPS5 applies rules in serial.

THE INTERACTION PROBLEM IN OTHER REPRESENTATIONS

Rule-based and logic-based representations are fairly similar with respect to the interaction problem. Like rules, logic provides for a direct way for drawing conclusions from situations. In the context of a specific problem, it is useful to encode only those propositions that can make problem-relevant conclusions. Logic-based representations are also like rules with respect to implementing special structures and dealing with different inference strategies. To implement R1 in predicate logic, for example, a subtask construct would also have to be implicitly programmed. Two different inference strategies for logic, such as PROLOG and resolution theorem proving, are quite different to use.

The emphasis in frame representations is on describing the conceptual structure of the domain. However, different problems might need quite different conceptual structures. For example, classificatory problem solving (Gomez & Chandrasekaran, 1981; Clancey, 1985) in general needs a generalization hierarchy (hypothesis-subhypothesis), while routine design (Brown & Chandrasekaran, 1986) in general needs a structural hierarchy (component-subcomponent). Of course, one of Minsky's original intentions was to express the interaction between knowledge and inference strategies. For example, the idea of attaching various kinds of information to a frame is for controlling how the frame will be used.

EXPLOITING THE INTERACTION PROBLEM

The interaction problem will not go away no matter what representation is chosen. Every knowledge-based system will be developed, debugged, and maintained so its knowledge works with its inference strategy and so its knowledge in combination with its inference strategy solves a certain problem. It is not feasible to undertake an exhaustive study of a domain to acquire any and all the knowledge associated with that domain. It is important to realize that knowledge-based systems are powerful only when selected portions of domain knowledge, appropriately interpreted, are needed to solve problems.

Instead of trying to lessen the impact of the interaction problem, our research strategy has been to *exploit* it. We claim that different representations can be exploited in different ways and are thus more applicable to certain kinds of problems than others. This is where our theory of generic tasks comes in. Our intent is to propose types of problem solving in which the representation and the inference strategy can be exploited to solve certain kinds of problems. For a particular domain and problem, our intent is to encode a selected portion of domain knowledge into an efficient and maintainable problem solving structure.

The proposal

Intuitively one would think that diagnosis in different domains would have certain types of reasoning in common, and that design in different domains would also have certain types of reasoning in common, but that diagnostic reasoning and design problem-solving will be generally speaking different. For example, diagnostic reasoning generally involves malfunction hierarchies, rule-out strategies, setting up a differential, etc., while design involves device/component hierarchies, plans to specify components, ordering of plans, etc. However, the formalisms (or equivalently the languages) that have been commonly used for knowledge-based systems do not capture these distinctions. Ideally, diagnostic knowledge should be represented by using the *vocabulary* that is appropriate for diagnosis, while design knowledge should have a vocabulary appropriate for design. Our approach to this problem has been to identify *generic tasks*—basic combinations of knowledge structures and inference strategies that are powerful for dealing for certain kinds of problems. The generic tasks provide a vocabulary for describing problems, as well as for designing knowledge-based systems that perform them.

CHARACTERIZATION OF A GENERIC TASK

Each generic task is characterized by information about the following:

(1) The type of *problem* (the type of input and the type of output). What is the function of the generic task? What is the generic task good for?
(2) The *representation* of knowledge. How should knowledge be organized and structured to accomplish the function of the generic task? In particular, what are the type of *concepts* that are involved in the generic task? What concepts are the input and output about? How is knowledge organized in terms of concepts? In essence, we adopt Minsky's idea of frames as a way to organize the problem solving process (Minsky, 1975).
(3) The *inference strategy* (process, problem-solving, control regime). What inference strategy can be applied to the knowledge to accomplish the function of the generic task? How does the inference strategy operate on concepts?

The phrase "generic task" is somewhat misleading. What we really mean is *an elementary generic combination of a problem, representation, and inference strategy about concepts*. The power of this proposal is that if a problem matches the function of a generic task, then the generic task provides a knowledge representation and an inference strategy that can be used to solve the problem. Figure 2 illustrates how this fits into knowledge acquisition. Identifying a generic task that is applicable to the problem is an intermediate goal of knowledge acquisition. The problem must match the type of problem that the generic task can solve. The generic task then specifies a type of strategy and a type (or form) of knowledge for solving the problem. Further stages of the knowledge-acquisition process (not illustrated in the figure) are to obtain domain knowledge and to particularize the inference strategy.

EXAMPLES OF GENERIC TASKS

Our group has identified several generic tasks. Here, we briefly describe the generic tasks of hierarchical classification (Gomez & Chandrasekaran, 1981) and object synthesis by plan selection and refinement (Brown & Chandrasekaran, 1986).

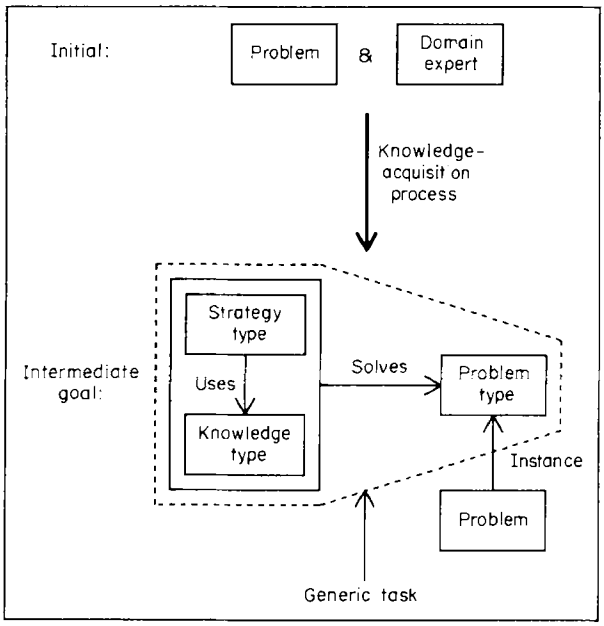

FIG. 2. Using generic tasks in knowledge acquisition.

Hierarchical classification

Problem: Given a description of a situation, determine what categories or hypotheses apply to the situation.

Representation: The hypotheses are organized as a classification hierarchy in which the children of a node represent subhypotheses of the parent. There must be knowledge for calculating the degree of certainty of each hypothesis.

Important concepts: Hypotheses.

Inference strategy: The establish–refine strategy specifies that when a hypothesis is confirmed or likely (the establish part), its subhypothesis should be considered (the refine part). Additional knowledge may specify how refinement is performed, e.g. to consider common hypotheses before rarer ones. If a hypothesis is rejected or ruled-out, then its subhypotheses are also ruled-out.

Examples: Diagnosis can often be done by hierarchical classification. In planning, it is often useful to classify a situation as a certain type, which then might suggest an appropriate plan. The diagnostic portion of MYCIN (Shortliffe, 1976) can be thought of as classifying a patient description into an infectious agent hierarchy. PROSPECTOR (Duda, Gaschnig & Hart, 1980) can be viewed as classifying a geological description into a type of formation. Hierarchical classification is similar to the refinement part of Clancey's heuristic classification (Clancey, 1985).

Object synthesis by plan selection and refinement

Problem: Design an object satisfying specifications. An object can be an abstract device, e.g. a plan or program.

Representation: The object is represented by a component hierarchy in which the

children of a node represent components of the parent. For each node, there are plans that can be used to set parameters of the component and to specify additional constraints to be satisfied. There is additional knowledge for selecting the most appropriate plan and to recover from failed constraints.

Important concepts: The object and its components.

Inference strategy: To design an object, *plan selection and refinement* selects an appropriate plan, which, in turn, requires the design of subobjects in a specified order. When a failure occurs, failure handling knowledge is applied to make appropriate changes.

Examples: Routine design of devices and the synthesis of everyday plans can be performed using this generic task. The MOLGEN work of Friedland (Friedland, 1979) can be viewed in this way. Also R1's subtasks (McDermott, 1982) can be understood as design plans.

OTHER PROPERTIES OF GENERIC TASKS

Generic tasks have a number of other properties. First, there is not a one-to-one relationship between generic tasks and problems. A problem might match the function of more than one generic task, so that several strategies might be used to solve the problem, depending on the knowledge that is available. For example, a design problem might be solvable by hierarchical classification if there is only a limited number of possible designs. Also, generic tasks can be composed for more complex reasoning, i.e. one generic task problem solver might call upon another to solve a subproblem. Typically, knowledge-based systems must be analysed as using combinations of generic tasks, rather than just using a single one.

Second, we are interested in human-like problem-solving, which, we assume, is about concepts and is non-quantitative. This leads us to look for ways that knowledge can be distributed over the concepts of a domain. Distribution of knowledge opens the possibility for parallel processing, e.g. refinement of a hypothesis in hierarchical classification can be done in parallel. The criterion of non-quantitative problem solving generally excludes normative methods such as linear programming and Bayesian optimization.

Third, a complete description of a generic task should include not only the items mentioned above, but also how it can be used for explanation, learning, and teaching. In this view, things like explanation and learning are not separate generic tasks, but are processes to be integrated into a generic task.

Fourth, each generic task can be associated with a programming language that embodies the kind of knowledge and inference strategy that the generic task specifies. For example, CSRL is a language developed for hierarchical classification (Bylander & Mittal, 1986), and DSPL, a language for object synthesis using plan selection and refinement (Brown, 1985). The languages of the generic tasks should be useful for guiding and mediating knowledge acquisition, which implies that each language should be useful for: pointing out what knowledge needs to be obtained, decomposing problems into subproblems, combining information in decision-making (especially uncertain information), debugging knowledge, allowing incremental expansion of the knowledge base, and providing process guidance to implementors.†

† This list is adapted from (Bradshaw, 1986).

OTHER GENERIC TASKS

Other generic tasks that have been identified include knowledge-directed information retrieval (Mittal, Chandrasekaran & Sticklen, 1984), abductive assembly of explanatory hypotheses (Josephson, Chandrasekaran & Smith, 1984), hypothesis matching (Chandrasekaran, Mittal & Smith, 1982), and state abstraction (Chandrasekaran, 1983). More detail on the overall framework can be found in Chandrasekaran (1986).

Exploiting hierarchical classification

Each generic task exploits domain knowledge differently; it calls for knowledge in a specific form that can be applied in a specific way. Because the knowledge-acquisition methodology must be able to extract and select the appropriate knowledge, each generic task calls for a different knowledge acquisition methodology. For illustration we consider the generic task of hierarchical classification (HC). In HC, the emphasis is on obtaining the classification hierarchy that contains the hypotheses that are relevant to the problem and can be used with the establish–refine strategy. This section does not provide a complete knowledge-acquisition methodology for HC, but outlines a number of considerations that a methodology must take into account. Additional guidelines for using HC can be found elsewhere (Mittal, 1980; Bylander & Smith, 1985).

DETERMINING HYPOTHESES OF INTEREST

HC is useful for determining the hypotheses that apply to a situation. An important step then is to decide upon the hypotheses that the problem-solver should potentially output. For example in diagnosis, the potential malfunctions of the object should be considered. The goal here is to determine the specific categories that should be produced, so if a general category is considered (e.g. "something is wrong with X"), then more specific categories should be generated (e.g. by asking "What types of problems can occur with X?"). Determining the usefulness of a category is discussed below.

ANALYSING COMMONALITIES AMONG HYPOTHESES

Once a collection of classificatory hypotheses have been identified, one needs to determine the commonalities among the hypotheses. These commonalities become potential candidates for mid-hierarchy hypotheses in the classification hierarchy. The easiest example to handle is when one hypothesis is clearly a subhypothesis of another, i.e. it asserts a more specific category. In general, two hypotheses may have commonalities along the following lines:

Definitional—The two hypotheses share a definitional attribute, e.g. hepatitis and cirrosis are liver diseases. Rain and snow are forms of precipitation;
Appearance—The two hypotheses are recognized using common pieces of evidence. Both cholestasis and hemolytic anemia have jaundice as a common symptom. Wet grass is symptomatic of both rain and dew;
Planning—The two hypotheses are associated with similar plans of action. Both

the common cold and allergies are reasons to take plenty of facial tissue with you. Either lightning or strong winds are good reasons for staying inside.

The ideal hypothesis asserts some definitional attribute over all its subhypotheses, has an appearance common to all its subhypotheses, and also provides constraints on the plans associated with its subhypotheses.

In general, the hierarchy should follow a definitional decomposition whenever possible. However, there are cases where appearance is an important consideration. For example, the Dublin–Johnson syndrome is a benign hereditary disorder that mimics key symptoms of cholestasis (jaundice, conjugated hyperbilirubinemia—high amounts of conjugated bilirubin in the blood). Because it *looks* so much like cholestasis, it is most useful to make it a subhypothesis of cholestasis.

ASSESSING EVIDENCE FOR OR AGAINST HYPOTHESES

The above two steps should generate a large number of hypotheses. However, not all of them will be useful for HC, i.e. there is a need to select a classification hierarchy that can be used to exploit the establish–refine strategy, getting rid of any intermediate hypothesis do not provide additional problem-solving power. Because the language we have used for HC, CSRL, requires a classification *tree* (Bylander & Mittal, 1986), we have become familiar with some of the strategies for evaluating hypotheses. However, the following questions are relevant whether a tree or tangled hierarchy is used.

Are there sufficient criteria to distinguish the hypothesis from other hypotheses? In other words, does this hypothesis have a different appearance from other hypotheses?

Is there evidence that distinguishes the hypotheses from its siblings? Because the establish–refine strategy does not consider a hypotheses unless its parent (or one of its parents in a tangled hierarchy) is relevant, evidence that distinguishes the hypothesis from its siblings is especially important.

Is the evidence normally available? Evidence for or against an hypothesis is not very useful if it is not likely to be available to the system when it is running. For example in medical diagnosis, some tests are relatively risky, expensive, or time-consuming to perform, so it is best to use hypotheses that rely on outward signs and symptoms and generally available laboratory data.

We have generally used another generic task, hypothesis matching, for mapping evidence to confidence values in hypotheses (Chandrasekaran *et al.*, 1982). However, we do not want to complicate the central issue by considering combinations of generic tasks. Examples of how hypothesis matching can be exploited are provided in Sticklen, Chandrasekaran & Smith (1985) and Bylander & Mittal, (1986).

DEBUGGING HYPOTHESES

An important part of knowledge acquisition is being able to find out what knowledge was incorrect or left out when something goes wrong. In HC, the following problems can occur:

Missing hypothesis—add the hypothesis to the classification hierarchy;

Wrong confidence value—debug the knowledge that produces the confidence value. Sticklen *et al.* (1985) describes how hypothesis matching can be debugged. The problems below assume that the confidence values are reasonable in view of the evidence considered;

Relevant hypothesis not considered—a hypothesis is not considered if one of its ancestors is not refined.† There are two possible problems with the ancestor.

There is not enough evidence to support the ancestor. To resolve this problem, one needs to find more evidence for the ancestor, lower the threshold for refining the ancestor, or implement more suitable ancestors, i.e. find better hypotheses for the establish–refine strategy.

The hypothesis is not definitionally a subhypothesis of the ancestor. In this case, the solution is to implement more suitable ancestors.

Irrelevant hypothesis considered—A hypothesis is considered only if one of its parents was refined.‡ Two causes of this problem are similar to the previous problems—when there is not enough evidence to oppose the parent or the hypothesis is not definitionally a subhypothesis of the parent. Similar fixes apply to these cases.

Another possible cause is that the establish–refine strategy being used is too simple. Sometimes a hypothesis should not be considered even if its parent is established. For example, if one of the hypothesis's siblings is confirmed, and the hypothesis is incompatible with its siblings, then the hypothesis should not be considered. The solution here is to adapt the establish–refine strategy to take this additional information into account. It should be noted that this problem is not a defect of establish–refine. Instead, it shows that establish–refine is really a *family* of strategies. The CSRL language, for example, provides a default establish–refine strategy and allows other establish–refine strategies to be defined.

KNOWLEDGE ACQUISITION FOR HIERARCHICAL CLASSIFICATION

The point of HC is to determine the hypotheses that describe a situation. The point of knowledge acquisition for hierarchical classification is to obtain the knowledge (the classification hierarchy) so that HC can be effectively performed. That is, knowledge acquisition needs to exploit the interactions between the representation, inference strategy, and the problem. Exploiting HC means the construction of a classification hierarchy that contains the hypotheses to be considered and that allows the establish–refine strategy to efficiently search the hypotheses. Thus a knowledge acquisition methodology for HC needs to evaluate each hypothesis for its relevance as a potential output and in view of the evidence that can support or oppose it.

A re-examination of past beliefs

Some generally held beliefs about knowledge-based systems need to be re-examined in light of the interaction problem and our proposal to exploit it. These beliefs have

† This statement is true only for a classification tree. For a tangled hierarchy, a hypothesis is not considered if every path from the root node to the hypothesis has a node that was not refined.

‡ It is possible that some other problem solver might directly ask a classifier to consider a hypothesis. This problem would be then attributable to the other problem solver, not the classifier.

served the first generation of knowledge-based systems well, especially in stimulating much research and discussion. However, we believe it is the time to reconsider them.

BELIEF 1: KNOWLEDGE SHOULD BE UNIFORMLY REPRESENTED AND CONTROLLED

This belief denies the interaction problem and implies that there is nothing to be gained by using different representations to solve different problems. Our experience is that when the problems of a domain match the generic tasks, the generic tasks provide explicit and powerful structures for understanding and organizing domain knowledge.

BELIEF 2: THE KNOWLEDGE BASE SHOULD BE SEPARATED FROM THE INFERENCE ENGINE

This belief denies that the inference strategy affects how knowledge is represented. However, its real effect has been to force implementors to implicitly encode inference strategies within the knowledge base. Both MYCIN, whose diagnostic portion is best understood as HC, and R1, which is best understood as routine design, show that this separation is artificial.

BELIEF 3: CONTROL KNOWLEDGE SHOULD BE ENCODED AS METARULES

Although metarules address the problem of how to have multiple, explicit strategies in a rule-based system, the metarule approach ignores other aspects of the interaction problem. The "separation of control knowledge from domain knowledge" promotes the view that domain knowledge can be represented independent of its use, i.e. that different sets of metarules can be applied as needed. However, given a clear strategy (whether metarules or inference engine) and a problem to be performed, the domain knowledge will be adapted to interact with the strategy to solve the problem.

BELIEF 4: THE ONTOLOGY OF A DOMAIN SHOULD BE STUDIED BEFORE CONSIDERING HOW TO PROCESS IT

We believe that ontology should not be performed just for its own sake, but in view of the problems that need to be done. For example, to apply HC to a domain, there is a need to focus on the hypothesis space and evaluate hypotheses. Although other knowledge structures (e.g. component hierarchies, causal networks) may be useful for other generic tasks, if HC is going to be performed, then knowledge acquisition should concentrate on those aspects of the domain that are relevant to HC. This is not to say that a domain should not be analysed to identify what generic tasks are appropriate; however, this kind of domain analysis does not require an exhaustive ontology of the domain.

BELIEF 5: CORRECT REASONING IS A CRITICAL GOAL FOR KNOWLEDGE-BASED SYSTEMS

Everything else being equal, being correct is better than being incorrect. However, an emphasis on correctness detracts from more critical issues. One of those issues is developing an understanding of the appropriate strategies to be applied to a

problem. For example, there has been much research and debate about normative methods for calculating uncertainty. The reasoning problem, though, is not how to precisely calculate uncertainty, but how to avoid doing so. In diagnosis, for example, there is much more to be gained by using abduction (assembling composite hypotheses to account for symptoms), then by independently calculating the degree of certainty of each hypothesis to several decimal places of accuracy.

BELIEF 6: COMPLETENESS OF INFERENCE IS A CRITICAL GOAL FOR KNOWLEDGE-BASED SYSTEMS

Everything else being equal, being complete is better than being incomplete, but an emphasis on completeness ignores the fact that certain kinds of inferences will be more important than others for a particular problem. For example in our description of HC, we did not mention that when a subhypothesis is confirmed, one can infer that its ancestors are also confirmed. This is not because we believe that a HC problem solver should never perform this inference, but because other inferences are the crucial aspects of HC: refinement of a hypothesis if it is likely and pruning of its subhypotheses when it is ruled out.

BELIEF 7: A REPRESENTATION THAT COMBINES RULES, LOGIC, FRAMES, ETC. IS WHAT IS NEEDED

Such representations appear to be a good compromise since they let you represent knowledge in the "paradigm" of your choice. Unfortunately, this is, at best, only an interim solution until something better is found. None of the individual representations fully address the interaction problem, nor do they distinguish between different types of reasoning.

Generic tasks at the "right" level of abstraction

The first generation of research into knowledge-based systems has conducted an extensive search for a "holy grail" of representation, in which knowledge could be represented free of assumptions of how it would be used. For any particular problem, though, certain kinds of inferences and certain pieces of knowledge will be critical to the problem, and consequently, domain knowledge needs to be organized so those inferences are performed efficiently. This is how the interaction problem arises, and why it will never go away. Instead of futilely trying to avoid it, the interaction problem needs to be studied and understood so that methods of exploiting it can be discovered and applied.

Our theory of generic tasks is an attempt to provide the "right" level of abstraction for this and other problems of knowledge-based reasoning. Each generic task provides a knowledge structure in which knowledge can be organized at a conceptual level. In hierarchical classification, the concepts are hypotheses organized as a classification hierarchy. Each generic task identifies a combination of a problem definition, representation, and inference strategy that exploits the interaction problem. We have shown how the generic task of hierarchical classification can be associated with a knowledge-acquisition methodology that takes advantage of the interactions between domain knowledge and the establish-refine strategy.

Research supported by Air Force Office of Scientific Research, grant 82-0255, and Defense Advanced Research Projects Agency, RADC Contract F30602-85-C-0010

References

BRADSHAW, J. M. (1986). Presentation at the *Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banff, Canada.

BROWN, D. C. (1985). Capturing mechanical design knowledge. *Proceedings of the 1985 ASME International Computer in Engineering Conference,* Boston.

BROWN, D. C. & CHANDRASEKARAN, B. (1986). Knowledge and control for a mechanical design expert system. *Computer,* **19,** 92–100.

BYLANDER, T. & MITTAL, S. (1986). CSRL: A language for classificatory problem solving and uncertainty handling. *AI Magazine,* **7,** 66–77.

BYLANDER, T. & SMITH, J. W. (1985). Mapping medical knowledge into conceptual structures. *Proceedings of Expert System in Government Symposium. IEEE Computer Society,* McLean, Virginia, pp. 503–511.

CHANDRASEKARAN, B. (1983). Towards a taxonomy of problem solving types. *AI Magazine,* **4,** 9–17.

CHANDRASEKARAN, B. (1984). Expert systems: matching techniques to tasks. In *Artificial Intelligence Applications for Business.* Norwood, New Jersey: Ablex, pp. 116–132.

CHANDRASEKARAN, B. (1986). Generic tasks in knowledge-based reasoning: high-level building blocks for expert system design. *IEEE Expert* **1,** 23–30.

CHANDRASEKARAN, B., MITTAL, S. & SMITH, J. W. (1982). Reasoning with uncertain knowledge: the MDX approach. *Proceedings of the Congress of American Medical Informatics Association.* San Francisco: AMIA, pp. 335–339.

CLANCEY, W. J. (1985). Heuristic classification. *Artificial Intelligence,* **27,** 289–350.

DUDA, R. O., GASCHNIG, J. G. & HART, P. E. (1980). Model design in the prospector consultant system for mineral exploration. In *Expert Systems in the Microelectronic Age.* Edinburgh University Press, pp. 153–167.

FORGY, C. L. (1981). Technical Report CMU-CS-81-135, *OPS5 Users Manual.* Carnegie–Mellon University.

FRIEDLAND, P. (1979). Knowledge-based experiment design in molecular genetics. *Ph.D. thesis,* Computer Science Department, Stanford University.

GOMEZ, F. & CHANDRASEKARAN, B. (1981). Knowledge organization and distribution for medical diagnosis. *IEEE Transactions on Systems, Man and Cybernetics,* **SMC-11,** 34–42.

JOSEPHSON, J. R., CHANDRASEKARAN, B. & SMITH, J. W. (1984). Assembling the best explanation. *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems.* IEEE Computer Society, Denver, pp. 185–190.

MARR, D. (1982). *Vision.* W. H. Freeman.

MCDERMOTT, J. (1982). R1: A rule-based configurer of computer systems. *Artificial Intelligence* **19,** 39–88.

VAN MELLE, W. (1979). A domain independent production-rule system for consultation programs. *Proceedings of the Sixth International Conference on Artificial Intelligence.* Tokyo, pp. 923–925.

MINSKY, M. (1975). A framework for representing knowledge. *The Psychology of Computer Vision.* McGraw–Hill, p. 211–277.

MITTAL, S. (1980). Design of a distributed medical diagnosis and database system. *Ph.D. thesis,* Department of Computer and Information Science, The Ohio State University.

MITTAL, S., CHANDRASEKARAN, B. & STICKLEN, J. (1984). Patrec: a knowledge-directed database for a diagnostic expert system. *Computer* **17,** 51–58.

SHORTLIFFE, E. H. (1976). *Computer-Based Medical Consultations: MYCIN.* New York: Elsevier.

STICKLEN, J.,, CHANDRASEKARAN, B. & SMITH, J. W. (1985). MDX-MYCIN: the MDX paradigm applied to the MYCIN domain. *Computers and Mathematics with Applications,* **11,** 527–539.