Machine Learning, 4, 339-345 (1989)

Task-Structures, Knowledge Acquisition and Learning

B. CHANDRASEKARAN CHANDRA@CIS.OHIO-STATE.EDU
Laboratory for AI Research, Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210

Abstract. One of the old saws about learning in AI is that *an agent can only learn what it can be told,* i.e., the agent has to have a vocabulary for the target structure which is to be acquired by learning. What this vocabulary is, for various tasks, is an issue that is common to whether one is building a knowledge system by learning or by other more direct forms of knowledge acquisition. I long have argued that both the forms of declarative knowledge required for problem solving as well as problem-solving strategies are functions of the *problem-solving task* and have identified a family of *generic tasks* that can be used as building blocks for the construction of knowledge systems. In this editorial, I discuss the implication of this line of research for knowledge acquisition and learning.

Key Words: Generic tasks, learning, knowledge acquisition, task-structure

1. Learning and Knowledge Acquisition

One of the more straightforward relations between knowledge acquisition and learning is that learning is one means of knowledge acquisition. But there is more to this relationship. One of the old saws about learning in AI is that *an agent can only learn what it can be told,* i.e., the agent has to have a vocabulary for the target structure which is to be acquired by learning. What this vocabulary is for various tasks is an issue that is common to whether one is building a knowledge system by learning or by other more direct forms of knowledge acquisition. For the last several years, my colleagues and I have been investigating *task-specific* architectures, in particular those that can support very general information-processing strategies that we have called *generic tasks* (GT's) [Chandrasekaran, 1986; 1987]. In this guest editorial, I would like to discuss the implication of this line of research for knowledge acquisition and learning.

Much of what I will say regarding the advantages for knowledge acquisition and learning of task-specific architectures is applicable not only to our particular work on generic tasks, but to other work in the task-specific spirit as well, e.g., the work of Marcus and McDermott [1989].

2. Generic Tasks

The GT view has been evolving, but the essence of the approach can be captured in the following ideas.

Problems, methods and subproblems. A problem (or a problem-solving goal) can have one or more *methods* associated with solving it (or achieving the goal). Each of the methods is characterized by forms of *knowledge* and *inference* that are necessary for carrying out the

method and by additional *subgoals* (or subproblems) that will need to be achieved (solved) in order to complete the application of the method for the problem. (A method can be a procedure where the sequencing of steps is all prespecified, but it can be more abstract; in Newell's problem space terminology [Newell, 1980], it can be a search in a problem space. In fact, such methods are the ones that are interesting from an AI point of view.)

For example, the problem of *classification* has a method called *hierarchical classification,* which consists of exploring the classification hypotheses organized as a hierarchy. This calls for knowledge in the form of *hierarchies* and inference methods which are variations of, and include as a default strategy, *top-down explorations* of the hierarchy. This method has subgoals in the form of *evaluating* the evidence for or against a hypothesis so that it can be established or rejected. This subgoal similarly can have many methods associated with it, each of which is characterized by its own knowledge and inference requirements.

GT's as problem/method/knowledge/inference packages. A combination of a problem/method/knowledge/inference structure is something that we have called a *generic task* (GT). An underlying assumption of the GT view is that there are a number of generally useful GT's which serve as subgoals or subproblems for *many* complex knowledge-rich problem-solving tasks. For example, hierarchical classification is an ubiquitous method in diagnosis and selection problems. Similarly, *abstract plan instantiation and refinement* is an equally general method for parts of design or synthesis tasks. *Hierarchical abstraction of data* is another very useful method for concept matching or recognition.

A high level language for each GT. The GT approach has identified a number of such generally useful problem/method/knowledge/inference combinations and has made tools available that can support each such GT and combine them as building blocks in the solution of more complex problems. The support is provided in the form of knowledge and inference primitives appropriate for each method, in terms of which domain knowledge and inference can be directly encoded.

GT's and knowledge acquisition. The GT view enables the knowledge engineer to associate problems with appropriate methods and seek out the knowledge and inference patterns that are needed to support the method and also to access other GT's, i.e., problem/method/knowledge/inference structures, that may be needed for any subproblems. Thus the GT view is a direct aid in knowledge acquisition, since it focuses the knowledge acquisition process on the operational knowledge and inference needs of the problem (or the task). How exactly the GT view helps in knowledge acquisition has been discussed at some length by Bylander and Chandrasekaran [Bylander and Chandrasekaran, 1987], but the idea is simple: the knowledge and inference primitives directly give the knowledge engineer a vocabulary in terms of which to seek both the declarative and control knowledge in the domain for the task.

Explanations of problem solving at the task level. The task-specific view in general and the GT view in particular give advantages by providing appropriate vocabularies in which explanations can be couched [Chandrasekaran, Tanner and Josephson, 1989]. Each of the methods can, in principle, explain its behavior by using the vocabulary of inference and control that is directly appropriate for it. When the right GT level tool is used to implement the method, the problem solver can "introspect" about its behavior at the level of abstraction that corresponds to that of the task. For example, the hierarchical classifier can explain its behavior using the language of establishing classificatory hypotheses and refining them,

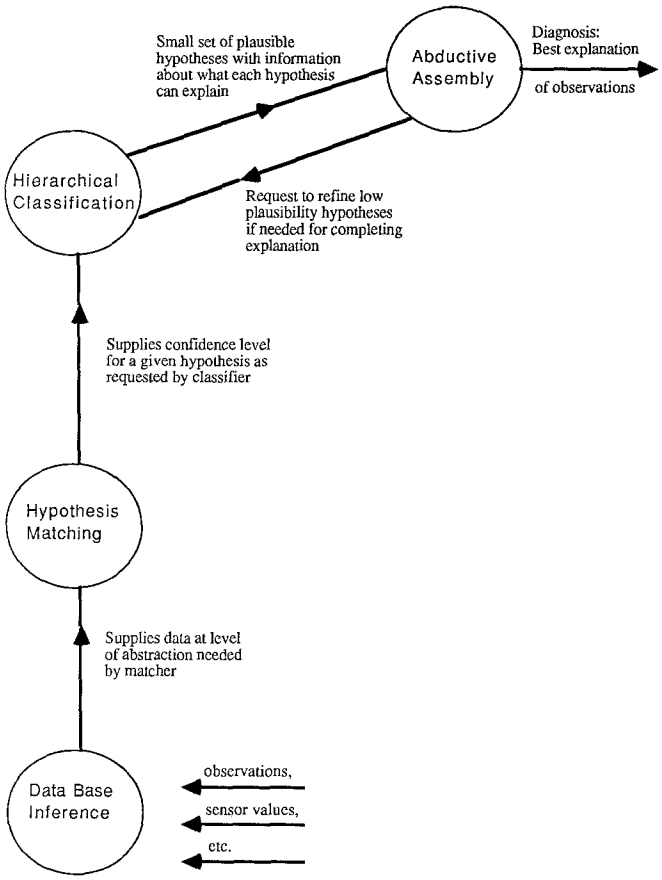

Figure 1. A generic task architecture for diagnosis with compiled knowledge.

and the plan instantiator and refiner can simply couch its behavior using a vocabulary of choosing between plans, invoking subplans, etc. If a diagnostic system is built using the generic tools for classification and hypothesis matching (see Figure 1), then the behavior of the diagnostic system can similarly be explained using the higher level explanatory vocabulary and a knowledge of the requirements of the task of diagnosis [Tanner, 1989].

3. Task Structures and Knowledge Acquisition

While I have talked in terms of generic tasks above, the points I am making are relevant for the more general notion of a *task-oriented methodology.* This methodology directs the process of analyzing and building knowledge-based systems for given problems by explicitly representing a *task-structure* for the problem[1]. This representation focuses the knowledge acquisition and system building stages by an explicit awareness of the requirements of the

tasks in the task structure. A task structure is a representation of the task in terms of the methods that are applicable for it in the domain and the conditions under which each method is applicable. Each method is itself specified in terms of how it uses knowledge and inference to achieve its goals, and in terms of what subgoals it sets up and requires to be achieved before it can succeed. This kind of decomposition can be done recursively until methods which achieve subgoals but which do not set up additional subgoals of their own are reached. This task structure has an enormous amount of leverage in directing knowledge acquisition and system building, since the knowledge and inference requirements for the methods can be explicitly identified. In the GT methodology, the task analysis is aided by the fact that a number of generic subgoals and methods have been identified; this repertoire provides guidance in the task decomposition.

For example, in Figure 1, a task-structure for diagnosis is given in terms of a number of generic tasks. In many domains an applicable method for diagnosis is that of assembling a best explanation. This method sets up the subgoals of *generating highly plausible hypotheses* and *abductive assembly of hypotheses into a best explanation.* Generating highly plausible hypotheses can be done by the method of hierarchical classification, which in turn sets up a subgoal of *evaluating a confidence level* for each of the hypotheses in the hypothesis space. An applicable method for this evaluation is *hypothesis matching* in which data are abstracted through a hierarchy of intermediate abstractions of evidence and finally into evidence about the hypothesis. Notice that this methodology permits us to choose methods for subgoals that can be supported by domain knowledge. In domain A, knowledge may be available for hypothesis evaluation by evidence abstraction as indicated, while in domain B the goal may best be accomplished by the method of Bayesian probability calculations because knowledge is available in the form of prior and conditional probabilities. The task structure makes explicit how goals are to be accomplished by what types of knowledge and inference. Knowing the type of knowledge needed for each method in turn can give focus to the knowledge acquisition process.

4. Explanation-Based Learning

The task-oriented view in general and the GT approach in particular have significant potential to aid learning. In addition to the knowledge-type vocabulary that they provide for each task, their ability to generate explanations at the right level can give significant leverage for learning. Explanation-based learning, broadly construed, is a method of learning by constructing an explanation of why some solution was correct or incorrect, and using the explanation to define the concept that is being learned. I would like to outline how the explanatory capabilities of the GT approach are helpful for learning by briefly describing a work in this vein that is being conducted in my laboratory by Bylander and Weintraub [1988].

The research attempts to build a knowledge-based system that performs corrective learning. When its answer to a problem is incorrect, the system attempts to identify which part of the knowledge it used for which task in the task-structure may be at fault and also attempts to change it. In order to explain how this is intended to work, a brief description of the theory of task-specific explanation described in [Chandrasekaran et al., 1989] may be useful.

We identify three types of explanation relating to knowledge-based systems. These are: (1) trace of run-time, data-dependent problem-solving behavior, i.e., explaining how the data in the current situation was used to arrive at a decision; (2) relating specific decisions to the control strategy used by the system; and (3) justifying particular pieces of knowledge by relating them to more general domain knowledge.

Let us use the example of hierarchical classification in diagnosis to make the above ideas a little clearer. Let us say that diagnosis is performed by performing hierarchical classification on the malfunction hierarchy (Figure 2) and that the control strategy is that of top-down refinement. If a hypothesis is established, its successors are examined; if a hypothesis is rejected, its subtree is pruned. Type 1 explanation will involve showing how a hypothesis was established by pointing to which data contributed evidence for and against it—there may be different ways of establishing the concept, and we want to know how in this particular instance the concept was established. Type 2 explanation would involve, e.g., explaining that a concept was rejected because its parent had been ruled out—this rejection being a consequence of the control strategy used in hierarchical classification. Let us say that one of the pieces of knowledge used in hierarchical classification is that data d1, d2, and d3 indicate a certain malfunction M in the hierarchy. Type 3 explanation will involve justifying this knowledge itself, e.g., by appealing to a structural model of the system that is being diagnosed and showing how malfunction M actually causes observations d1, d2, and d3.

The diagnostic and learning system being built by Weintraub and Bylander works in the domain of pathologic gait analysis. This diagnostic system combines abductive assembly and hierarchical classification problem solvers as in Figure 1 with a qualitative model of gait. The human expert, during the learning mode of the system, evaluates the correctness of the answer given by the system. If the system's answer is incorrect, the expert also provides the correct answer.

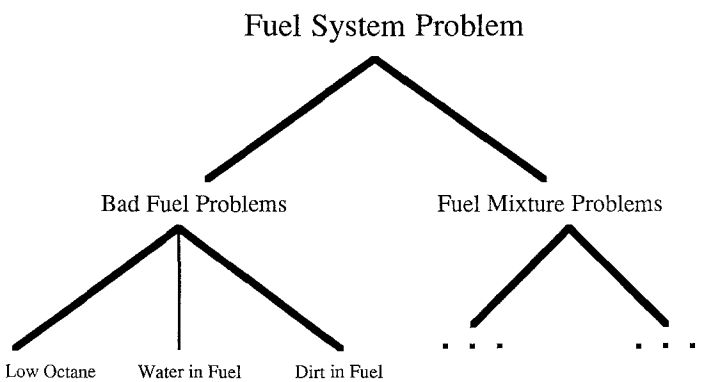

Figure 2. Example of a classification hierarchy.

Type 1 and Type 2 explanations can be constructed for the incorrect answer. Specifically, data used in support of bad judgments about individual hypotheses (Type 1) as well as how the control strategy led to errors (Type 2) can be constructed. Without getting into details, these explanations together can be used to identify possible knowledge elements that could have been responsible for the error, and which tasks in the task structure they are associated with. For example, the source of an incorrect decision may be traced to the fact that the evidence for a particular hypothesis is being incorrectly evaluated in the "Hypothesis Matching" subtask. Thus the explanation capability associated with the task-specific view helps solve some aspects of the *credit assignment problem* for learning.

The Type 3 explanation produced by the qualitative model can give additional pointers both for locating the possible places for the error as well as help in generating alternatives. Put another way, explanations of Types 1 and 2 help identify *what* is wrong, and Type 3 explanation will help identify *how* it is wrong. Thus the corrective learning is accomplished by a form of explanation-based learning, which in turn is made possible by the task-specific architectures that help specify how the task structure makes decisions.

5. Concluding Remarks

Understanding how learning can be used to build knowledge systems for problem solving involves uncovering both *general mechanisms* that play a role independent of the task or domain involved, e.g., *chunking* [Laird, Newell & Rosenbloom, 1987], as well as a specification of *what* needs to be learned for the problem at hand. Much of the interest in the field of knowledge systems has revolved around general, i.e., task-independent, architectures, which do not directly support distinctions in knowledge and inference for different types of tasks. There is a tendency to think of *all* knowledge as so much domain-specific details in this architecture, and hence to think that one cannot say interesting things, from an AI point of view, about learning this knowledge. On the other hand, we have discussed in this editorial task-specific but domain-independent information-processing strategies. A task-structure specifies how the problem at hand can be solved by using such generic tasks, and a theory of what knowledge needs to be acquired can be developed for these tasks. This level of abstraction directly helps in knowledge acquisition and, by providing explanations of problem solving that match the tasks, can help in learning.

Acknowledgments

The research reported is supported by Air Force Office of Scientific Report, grant 87-0090. I also thank Mike Weintraub and Tom Bylander for comments on an earlier draft.

Notes

1. A clarification of terms may be useful at this point. We have used the term "problem" to describe the task that a problem solving system is set, e.g., diagnosis, and the term "task" to refer to the more generic subproblems, e.g., classification, in the task-structure. This is for expository convenience and no absolute distinction is meant.

References

Bylander, T., and Chandrasekaran, B. 1987. Generic tasks for knowledge-based reasoning; The "right" level of abstraction for knowledge acquisition. *International Journal of Man-Machine Studies,* 26, 231-243.

Bylander, T. and Weintraub, M. 1988. A corrective learning procedure using different explanatory types. *AAAI Mini-Symposium on Explanation-Based Learning.* Stanford University, Palo Alto, CA.

Chandrasekaran, B. 1986. Generic tasks in knowledge-based reasoning; High-level building blocks for expert system design. *IEEE Expert, 1,* 23-30.

Chandrasekaran, B. 1987. Towards a functional architecture for intelligence based on generic information processing tasks. Invited talk, *Proceedings of the International Joint Conference on Artificial Intelligence.* IJCAI.

Chandrasekaran, B., Tanner, M. and Josephson, J. 1989. Explaining control strategies in problem solving. *IEEE Expert, 4,* 9-24.

Laird, J.E., Newell, A. and Rosenbloom, P.S. 1987. SOAR: An architecture for general intelligence. *Artificial intelligence, 33,* 1-64.

Marcus, S. and McDermott, J. 1989. SALT: A knowledge acquisition tool for propose-and-revise systems. *Artificial intelligence, 39,* 1-37.

Newell, A. 1980 Reasoning, problem solving and decision process: The problem space as a fundamental category. In L. Erlbaum (Eds.) *Attention and performance, VIII.*

Tanner, M. 1989. *Explaining knowledge systems: Justifying diagnostic conclusions.* Ph.D. thesis, Department of Computer and Information Science, The Ohio State University, Columbus, OH.