Models versus Rules, Deep versus Compiled, Content versus Form

Some Distinctions in Knowledge Systems Research

B. Chandrasekaran, Ohio State University

THE LITERATURE ON KNOWLEDGE system design makes four distinctions that require careful analysis: model-based versus rule-based problem solving, models versus experiential or associational knowledge, deep versus compiled knowledge, and content versus form. In discussing these distinctions I focus on the diagnostic task, since these distinctions have most often been debated in the context of this problem, but the issues are relevant to knowledge systems in general.

Models and rules

The "model-based versus rule-based" distinction mixes categories and hence does not set up genuine alternatives. All descriptions of any aspect of reality are models. Thus, Mycin,[16] MDX,[17] Neomycin,[18] and MDX2[19] are all model-based systems for diagnosis; that is, they all use bodies of knowledge that describe their domain, but the vocabularies that describe these models differ. We should distinguish the concepts for describing these models from their symbol-level representations.[14] "Rules" are one symbol-level commitment to represent models.

Almost invariably, the term "model" has been used to refer to one type of model, intended for diagnostic tasks. The device being diagnosed is usually described in terms of components, relations between components, and some sort of behavioral description of components. I call this class of models "SBF" models to indicate that their content deals with the structure, behavior, and function of device components.

Three distinct senses of the phrase "rule-based" were rolled into one in early discussions of knowledge-based systems:

(1) Rule systems as general-purpose programming languages — Computer science has generated several rule-based programming languages, including Post Production Systems, Markov Algorithms, the OPS family, and Emycin. These rule languages are especially useful in supporting a declarative style of knowledge representation. Many expert systems that were built using rules were simply algorithms implemented with a rule-programming language, without a commitment to any specific view of intelligence (human or artificial) or even knowledge.

IDENTIFYING KNOWLEDGE IN RULE FORM AS EXPERIENTIAL OR ASSOCIATIONAL IS COMMON BUT ERRONEOUS. IT MIXES UP THE ROLE OR ORIGIN OF KNOWLEDGE WITH ITS SYMBOL-LEVEL REPRESENTATION.

(2) Rule systems as human cognitive architectures — Newell and Simon proposed a type of rule-based architecture as a model for human cognition.[20] This is a computational-level model; that is, it lays out the data structures and operations by which computation proceeds. Rule-based AI systems derive some of their aura from this implied connection to cognitive architectures. However, to take a well-known system as an example, Mycin owes its success less to its use of the rule system as a cognitive model, and more to its use as a knowledge-programming language to implement a classification strategy.

(3) Rules as "rules of thumb," or heuristic associations — The rule-based expert system movement theorized that expertise in fact is comprised of a collection of heuristic associations. Mycin rules were supposedly of this type. Proponents also suggested that heuristic associations originate as empirical problem-solving experience. Expertise of this type was contrasted with so-called theory-based knowledge, which was supposed to be based on scientific principles underlying the domain, but not yet in a form useful for problem solving. Using rules in this sense really made a statement about the origin and character of the knowledge represented, not about rule systems as programming languages or cognitive architectures.

We face quite a potential for confusion when we fail to distinguish among all these senses of rules. In discussing the model-based versus rule-based distinction, for example, authors generally do not clarify when they are using rules as symbol-level representational ideas and when as empirical associations.

Because of rules being equated with empirical associations, some authors began to suggest that we needed "models" as opposed to "rules," where models stood for "first principles," then equated with "scientific laws." Another merging of ideas occurred: As rules were identified with empirical associations, "models" were associated with physics or first principles.

Consider a piece of knowledge that a diagnostician might use: "Excessive temperature at the outlet valve indicates that Valve A might be blocked." The diagnostician might have derived it by generating an account of the consequences of the broken valve, or stumbled on it by association. The fact that the diagnostician has this knowledge and uses it for diagnosis does not say anything about the origin of the knowledge — whether it was obtained associationally, experientially, or whatever. Conversely, representing $E = mc^2$ as a rule does not suddenly make it experiential or associational. Identifying knowledge in rule form as experiential or associational is common but erroneous; it mixes up the role

or origin of knowledge with its symbol-level representation. Equating expertise with rules and in turn with empirical association continues to be a source of confusion.

The diagnostic task requires knowledge that facilitates mapping from observations to diagnostic hypotheses. The piece of knowledge relating a certain temperature to a valve blockage is directly usable in the diagnostic task. It directly relates an observation to a malfunction hypothesis. During diagnostic problem solving, we need not generate nor access the causal pathway connecting the blockage and the high temperature.

This points to the importance of understanding the relation between tasks, methods, and types of knowledge. In an earlier article, I discussed the nature of task-specific knowledge for diagnosis and how that knowledge might be related to domain knowledge in other forms.[4] Are there other sources of knowledge from which diagnostically useful knowledge can be derived, or "compiled"? I suggested that such a knowledge source is likely to contain knowledge of the structure of the system being diagnosed. That article also related tasks to types of knowledge needed to accomplish them, and suggested that the idea of compilation was closely connected with the abstract task to be accomplished. While the term compilation has technical connotations in computer science as well as in common linguistic use, my sense of compilation — generating knowledge required to support a method for a task — is related to the common meaning of "gathering" or "collecting."

The diagnostic task

For the purpose of this discussion, we can identify a restricted class of diagnostic problems as follows. (I need to emphasize the word "restricted," since the following definition does not cover many important diagnostic problems.) We have a device comprised of components connected in a certain fashion. Information, energy, or physical fluids can flow through the device in a manner determined by the behavioral properties of the components and connections. The components might be recursively defined as devices themselves, down to some convenient or sufficient level for the purpose at hand. A set S of state variables is defined. (To avoid encumbering this discussion, I use the term "state" when I mean a partial state, often described by using a state variable.) A directly observable subset O of S is said to yield observations. A subset O_f of O is defined such that expected or normal values for observations in this set are known or can be inferred immediately. Deviations from these expected values trigger the diagnostic process.

A class of diagnostic problems for the device is specified by O, O_f, along with specification of expected values for observations in O_f, and a set C comprising causes of interest. For our restricted diagnostic problem, typical causes include malfunctioning components as well as input information or relations between components that are outside the specifications.

IF THE KNOWLEDGE NEEDED FOR A TASK OR METHOD IS DIRECTLY AVAILABLE, WE SAY THAT THE SYSTEM HAS COMPILED KNOWLEDGE.

An instance of the diagnostic problem in this class occurs when a given set of observations includes some deviation from expected values. The goal is to determine the subset of C that is the most likely to have caused the observations.

In general, O is a small subset of the set of all state variables of the device. We need to reason about other, unobservable states from the observations.

Compiled and deep knowledge

To keep this discussion concrete, I use the language of searching in problem spaces to discuss tasks and methods. Similar points can be made when viewing problem solving as inference making, constraint satisfaction, best-match retrieval, or other alternative formulations. Different tasks and methods need different kinds of knowledge. One definition is central to my approach: If the knowledge needed for a task or method is directly available, we say that the system has compiled knowledge.

A problem-solving task, or simply a task, is defined by characterizing the starting and desired end states of knowledge of the problem-solving process. For the diagnostic problem described earlier, we can state the task as follows:

- The starting state includes a set of observations of the system being diagnosed, where some of the observations deviate from the expected.
- The desired end state includes a subset of causes (for example, malfunctioning components or incorrect relations between components) that together caused the observations.
- A method is a proposal about how to organize information processing to achieve a goal. We can think of a method as a collection of problem spaces that are made available to achieve the task. We can describe a method in terms of the spaces of alternative partial solutions, the operators in each space that help generate candidate solutions, the objects that the operators operate on, and any additional knowledge about how to organize operators to achieve the task. Many methods might be available to achieve a given task.
- Knowledge is needed to apply method M to task T. We use this knowledge, say $K(T,M)$, to set up the problem spaces and apply operators to change knowledge states.

A common and simple method for the diagnostic task I've described is classification, applicable when the causes of malfunction do not interact. In this method, we map the observations onto the set of malfunction causes C. We compute some likelihood of each element in C, and propose hypotheses whose likelihoods exceed a given threshold as the answer. Two distinct types of problem solving occur. In one, we explore the space of malfunctions; in the other, we determine the likelihood of each malfunction and the explanatory coverage it offers.

To use the classification method for this version of the diagnostic task, we need the following types of knowledge:

- the set C of malfunction hypotheses, and

• the relationships between observations and malfunction hypotheses, so that likelihoods of various causes can be computed.

Depth of knowledge is relative. If the knowledge $K(T,M)$ needed by method M for task T is directly available to a knowledge system, we say that the knowledge system has the needed knowledge available in a compiled form (in the ordinary English sense of "collected").

If knowledge element k in $K(T,M)$ can be derived from another type of knowledge K' by additional reasoning or problem-solving activities, then K' is deep relative to k. Knowledge elements of K' itself might be derivable from yet another body of knowledge K'' by additional problem solving.

For the diagnostic example, consider a knowledge element $k(h,o)$, defined as knowledge that indicates whether a malfunction hypothesis h in C could have caused an observation o. Most first-generation diagnostic systems used some version of the classificatory method, and their knowledge bases contained this information in some form. For example, Mycin had rules that connected observations and hypotheses with degrees of certainty. If $k(h,o)$ is not directly available, it might be derivable from other types of knowledge.

Consider the following examples of derivation from other knowledge structures:

(1) Causal sequences of states. Knowledge in the form of a causal account of malfunction h consists of a sequence of states of interest linking a state and its causal consequence as the malfunction h propagates. Some of the states in the causal sequence might be observable. The causal account can have branches conditioned on other device parameters and inputs. This causal account can be traced to see if it includes observational states of interest. The functional representation organizes causal sequences by explicitly indicating the roles played by components and domain laws.[5] These causal accounts of malfunctions (or functions, for that matter) can themselves be derived from other sources of knowledge.

(2) A structure-function description of components. I call this an SBF model for a device. This model contains a description of device components, their I/O specifications, and composition knowledge that lets us derive I/O properties of various component configurations. If malfunction h is a specific component violating its I/O specifications in a particular way, we might derive a causal account (or parts of it) through simulation methods that use the SBF specification of components. We can then derive $k(h,o)$ from this causal sequence. As I said earlier, using SBF models in this manner for diagnosis has been called, somewhat imprecisely, model-based reasoning in diagnosis.

It is worth restating that the notion of depth of knowledge is relative, and no one body of knowledge can be said to be deep relative to a knowledge element k. In the above example, causal accounts and SBF models are both deep relative to $k(h,o)$. Bylander gives a somewhat different definition of depth,[21] but the underlying senses of depth converge — at least for the case of device diagnosis.

THE IMPORTANT RESEARCH ISSUES HAVE TO DO WITH HOW PROBLEM SOLVERS CAN MOVE FLEXIBLY FROM ONE MODEL TO ANOTHER IN THE PURSUIT OF GOALS.

Let's compare this notion with an aspect of the Soar architecture.[22] When Soar is missing knowledge needed to apply an operator in a problem space, it sets up a new problem space in which the needed knowledge can be generated. In the diagnosis example, the diagnostic hypothesis space is the original problem space, $k(h,o)$ is part of the knowledge needed to establish or reject a hypothesis h, and the causal knowledge structures are candidates for additional problem spaces to generate this knowledge.

Deep models are more robust. Suppose a piece of knowledge k can be derived from another piece of knowledge k' and vice versa. In this case, k and k' are deep models relative to each other, but that is not very interesting. In general, however, information is lost when one compiles a knowledge element from a deep model. Except in trivial problem spaces, discarding those portions of the space that do not help achieve the current goal involves loss of information. In my examples, the causal account of malfunction contains information about internal device states, which is missing in $k(h,o)$. SBF models contain information about normal component functioning that is missing in causal accounts of malfunctions. It is this unidirectionality of information retention that makes some knowledge types deep with respect to others.

Deep models are more robust in the sense that they are potentially applicable in a larger variety of problems than knowledge that is tuned for a task or a particular version of it. Suppose that, because of changes in domain technology (different probes, for example), some internal states that were not observable have been made observable, and vice versa. The device is the same, the malfunctions are the same, but the diagnostic task is now different. $K(T,M)$ for this new version of the task is correspondingly different. However, we can reprocess the causal accounts of malfunctions to obtain $k(h,o)$'s for building the new $K(T,M)$.

Suppose the device remains the same, but the purposes for which it is being used have changed. The set of malfunctions is no longer the same. Again, the diagnostic task has changed and, consequently, $K(T,M)$ differs as well. The causal accounts that had been generated for the previous set of malfunctions are no longer useful. However, we can still use the SBF models of the components and the simulation process to derive the causal accounts for the new malfunctions, and consequently to derive the elements of the new $K(T,M)$.

When to build compiled knowledge structures? We have two ways to generate the knowledge needed for a task. We can do so at runtime on an as-needed basis by accessing the portions of the deep knowledge structure that are relevant to the current problem, or we can build a large store of task-specific pieces at the beginning and keep them available. This issue is separate from the idea that tasks and methods determine the type of knowledge needed, which in turn can be derived from more general knowledge structures.

Nevertheless, when to compile and store

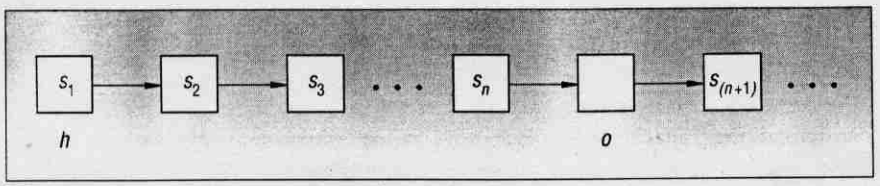


Figure 1. In Case 1, each state has a single causal consequence.

a knowledge element is an important question. This is really an instance of the more general question of when to "cache" the result of any problem solving. Three criteria spring to mind for caching:

- The problem-solving activity involved in generating the knowledge element requires an exponential amount of computation — for example, exponential search.
- The knowledge element generated is likely to be needed sufficiently often.
- The knowledge element can be generalized and indexed by the situations for which it is appropriate. A good vocabulary of tasks, methods, and the knowledge needed to support them eases the indexing problem.

Let's take a look at the nature of search in compilation. As discussed earlier, SBF models can help generate the two types of knowledge needed for the classification method: malfunction hypotheses, and knowledge of relations between observations and hypotheses. Here I consider only one of these types of knowledge, the knowledge element $k(h,o)$.

Case 1: Linear causal accounts. In Figure 1, $s_1,\ldots,s_n$, are unobservable internal states of the device, where s_1 is the hypothesis h, and each state gives rise to one relevant successor causal state. Whether to store "$h \rightarrow o$" depends on the value of n. The savings here are linear.

Case 2: Multiple consequences. Suppose each cycle of propagation produces more than one relevant successor state; for example, "leaky valve → reduced cooling water," and "leaky valve → shorting of controller circuit." As shown in Figure 2, some simulation paths converge on o while others go to states that are not observationally significant. At runtime, the diagnostic problem solver must perform a search, starting from h, to determine if h causes o. Depending on how branching at one node is related to branching choices taken at other nodes, the search process may be combinatorial, but it will be at least polynomial. In either case, compiling the knowledge "h → o" here can produce considerable savings during runtime.

For sufficiently complex systems, we have no guarantee of finding causal paths leading to the observation of interest (if they even exist) within a fixed time. This is because we don't know whether additional propagation of consequences will result in observational states of interest.

The complexity of search in the SBF model space is especially high when the device has several complex subsystems at different levels of abstraction, any one of which might be chosen for propagating effects. My favorite simple example is from the medical domain. One of the functions of the liver is to produce bile, which is deposited in the duodenum. There, it is broken down as it is used in the digestive process. When the bile duct is obstructed, one cycle of simulation leads to "bile duct obstruction (h) → no bile in the duodenum (o_1) and bile retained in the liver (o_2)." However, neither o_1 nor o_2 is easily observable. But, as it happens, due to complex interactions between liver and renal systems, "o_2 → direct bilirubin in urine (o_3)," where o_3 is observable. Even when SBF models for the various physiological subsystems are available, identifying causal paths such as "bile duct obstruction → bilirubin in urine" is in the nature of a small medical discovery. Such discoveries are prize possessions of the diagnostic community. Any physician who tries to perform simulations based on SBF models without access to such compiled diagnostic knowledge is unlikely to be successful.

Case 3: Interacting causes. A combination of malfunctions can cause a different set of observations than the union of their individual effects. This is a fundamental source of the complexity of diagnostic reasoning for multiple faults. Since in theory we need to consider all combinations of malfunctions, the total amount of work in the simulation space can be substantial. In fact, most AI diagnostic systems make arbitrary cutoffs in the number of malfunctions to be jointly considered, the single fault assumption being the most common.

If medicine is any guide, human problem solvers have a rich precompiled store of lists of malfunction combinations that can interact or be causally related. During diagnostic reasoning, the problem solver might first try to explain the data by considering only the combinations in this list.

Combining models. The above analysis results in a dual message. On one hand, diagnostic knowledge that helps map from observations to causes often requires extensive searching in the space of causal states generated from SBF models, so caching those relations is a good idea. On the other hand, especially in cases 2 and 3 above, no body of compiled diagnostic knowledge can be guaranteed to be complete, except in the case of simple devices, since the simulator cannot presume whether additional propagation will result in an observation of interest. In fact, SBF models of devices are themselves unlikely to be complete. Before constructing the structure/behavior model of a device, the designer makes choices about the behaviors of interest that will be represented, and a diagnostic problem might have observations that require simulation of additional modalities of behavior not explicitly represented. This situation is at the root of the empirical fact that almost all nontrivial human diagnostic reasoning uses a combination of methods — accessing compiled diagnostic knowledge as well as resorting to various SBF and other models.

Diagnostic strategies usually combine these methods in several ways. For example:

(1) When knowledge for evaluating a hypothesis is missing, an appropriate version of an SBF model might be accessed selectively — in the sense that only the parts thought relevant to the current hypothesis are invoked — to derive the needed knowledge. Sticklen's MDX2 system uses functional models of physiological subsystems to compile the knowledge elements needed for specific diagnostic hypotheses.[19]

(2) A common diagnostic strategy starts by using reasonably complete diagnostic knowledge structures to perform diagnosis. If all the data are explained satisfactorily,

the process stops. If not, SBF or other models are selectively accessed to look for causal pathways by which the current set of hypotheses can explain the unexplained data as well, or other hypotheses can be identified. A variant of this strategy occurs when the hypotheses explain all the data but the problem solver thinks that causal connections between hypotheses might produce a more satisfying and parsimonious diagnosis. To try to establish causal connections, the problem solver can selectively search SBF models, especially those richly indexed with information about function and behavior at different levels of abstraction. Punch has adopted this strategy.[23]

(3) Human diagnosticians often try to convince themselves that a diagnosis they have arrived at makes sense by constructing causal stories to support the diagnosis. A knowledge system might first perform a diagnosis by means of a compiled knowledge structure, and then try to build a detailed causal explanation from the diagnoses to the observations by using SBF or other causal models. Using this strategy, Keuneke generates diagnostic explanations from functional representations of process systems.[24] The fact that the diagnosis has strongly hypothesized a causal path and its direction radically reduces the search in the simulation space.

Form and content in knowledge-based problem solving

Does compilation change the content of a knowledge base? If one learns a fact that is simply a logical consequence of other facts, has there been a change in the content of one's knowledge? The distinction between form and content is a subject of ongoing debate in AI, but the terms are poorly defined or understood. In one sense of the word "content," a set of axioms has the same content as the set of all theorems that are derivable from it. But that is scant comfort for a problem solver trying to prove theorems in that axiom system. No inferential or problem-solving activity adds content in this sense; in fact, throwing away details of the search and retaining only the successes reduces content. But just as having an appropriate lemma might make proving theorems easier, directly having access to diagnostic knowledge will help in diagnostic problem solving. This is a form of operationalizing knowledge that has also been discussed in the research on explanation-based learning.

Two senses of the word "form" are often confused in discussions on knowledge systems. One sense has to do with the vocabulary in which the knowledge needs to be expressed for a certain task. For example, we say that the classification method for the diagnostic task needs knowledge in a form that relates malfunctions to observations. This sense of form actually refers to the kinds of knowledge needed, that is, the kinds of things represented. For example, diagnosticians need knowledge of could-be-caused-by relationships for many diagnostic methods. Form in this sense is thus, paradoxically, a statement about the content of knowledge. The other sense of "form" has to do with the symbol-level representation adopted: whether the knowledge is represented in the form of rules, frames, sentences in predicate calculus, or whatever. I have argued for several years that there has been entirely too much concern with form in the latter sense, and not enough with form in the former sense.

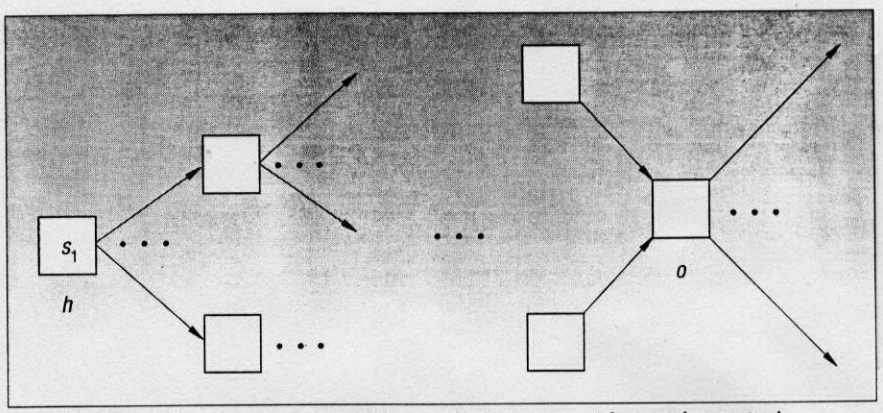


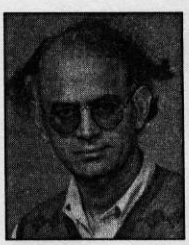
Figure 2. In Case 2, each state has multiple causal consequences (the initial state is the hypothesis).

IN THE LONG RUN, THE TERMS we've used are not as important as the distinctions they try to capture. The intuitions that underlie my work relate to the idea that a close relation exists between tasks, methods, and types of knowledge needed for methods. The knowledge needed for some types of tasks is itself the result of problem solving using other types of knowledge. These intuitions have formed the basis of work by me and my colleagues on generic task architectures[25] and, more recently, on task structures.[26] These intuitions also motivate my notions of depth and compilation.

In my view, the important research issues have to do with how problem solvers can move flexibly from one model to another in the pursuit of goals. We have discussed in this article how causal models and more compiled methods can be combined in diagnostic reasoning. Task structures can play a role in providing a flexible integration of these models.

Acknowledgments

This research was supported under DARPA contract F-49620-89-C-0110 and AFOSR grant 890250. I also owe thanks to Tom Bylander, Matt DeJongh, Tom Dietterich, Richard Fox, Ashok Goel, Yumi Iwasaki, John Josephson, Dale Moberg, Jack Smith, Jon Sticklen, and Mike Weintraub for useful comments on early drafts.



B. Chandrasekaran directs the AI group and is professor of computer and information science at Ohio State University. He is also editor-in-chief of *IEEE Expert*. He received his BE with honors from Madras University in 1963 and his PhD from the University of Pennsylvania in 1967. His current research interests address knowledge-based reasoning.

The author can be reached at 217 Bolz Hall, Ohio State University, 2036 Neil Ave., Columbus, Ohio 43210; e-mail, chandra@tut.cis.ohio-state.edu