CHARACTERIZING KNOWLEDGE DEPTH IN INTELLIGENT SAFETY SYSTEMS

TIM FININ
Paoli Research Center, Unisys Corporation, Paoli, Pennsylvania 19301

DAVID KLEIN
IBM T. J. Watson Research Center, Yorktown Heights, New York 10598

Intelligent process control may be viewed as encompassing four major tasks. An intelligent agent must monitor *the target system to obtain the values of relevant state variables in order to detect problems and to ascertain the status of the components that may be employed in responding to those problems. An intelligent agent must* determine plans *for managing the current situation. An intelligent agent must select a response (the "best" one) through a process of* plan evaluation. *Finally, to carry out the chosen response, the agent must perform* plan execution. *While monitoring and execution are relatively straightforward operations, plan determination and plan evaluation may be accomplished in a number of ways that vary in their relative depth of reasoning. In this paper we sketch an analysis for the reasoning underlying plan determination and evaluation tasks for a class of intelligent control systems that attempt to "provide a safety function." This analysis has two objectives: to illustrate a domain-independent mode of analysis for examining progressively deeper models, and to make the analysis available to those interested in building systems that provide safety functions.*

INTRODUCTION

Several authors have noted the distinction between "deep" and "shallow" models of expertise in expert systems (Hart, 1982; Chandrasekaran and Mittal, 1983; Fink, 1985). By shallow models we usually mean that conclusions are drawn directly from observed facts that characterize a situation. An advantage of shallow models is that they often directly encode the heuristics that experts use in performing their reasoning tasks, and they are thus relatively easy to build. In addition, shallow models tend to be relatively efficient because they select rather than construct their solutions. One disadvantage of shallow models, however, is that explicitly stating all the preconditions under which a solution should be selected is an error-prone process. Another weakness of shallow models is that they are inflexible, unable to deal with circumstances even slightly different from those explicitly anticipated (de Kleer and Brown, 1984). In addition, shallow models may be difficult to maintain, since what is conceptually a single

Copyright © 1989 by Hemisphere Publishing Corporation

piece of knowledge may be unsystematically distributed across several objects in a knowledge base. Finally, explanations generated from shallow models tend to be limited to traces of the inference chains that lead to conclusions.

In contrast, deep models of expertise correspond more closely to the notion of reasoning from first principles. They tend to be more robust than shallow models, handling problems not explicitly anticipated and exhibiting higher performance at the periphery of their knowledge. Deep models can also be more easily analyzed for correctness and completeness. For example, in device-centered models of physical systems (de Kleer and Brown, 1984; Davis, 1984), each physical device maps directly into a structured object in the representation. Deep models of expertise are also more useful for generating explanations in that reasoning steps that are usually implicit in shallow models can be elucidated. Deep reasoning is, however, bound to be slower and more complex than shallow reasoning because a more sophisticated control structure is required (Koton, 1985).

Abstractly characterizing deep and shallow models and contrasting their relative merits in a general way provides little direction for knowledge engineering. In particular, the field lacks a definition of exactly what makes a model "deep" and lacks guidelines regarding the appropriate depth of models for a given application. We believe that such guidelines should be developed by abstracting from a large set of examples. In particular, we advocate the approach of (1) adopting an informal definition of "knowledge depth," (2) isolating high-level reasoning tasks (e.g., diagnosis, simulation) for analysis, and (3) for each such reasoning task, contrasting the merits of models inspired by domains that vary in their relative depth.

In this paper we sketch such an analysis for a reasoning task called "provide a safety function" (which is defined later) in order to achieve two objectives: to illustrate a domain-independent mode of analysis for examining progressively deeper models, and second, to make the analysis available to those interested in building systems that provide safety functions.

The paper is organized as follows. The next section provides a simple operational definition of depth that is used in the ensuing analysis. We then define the reasoning task "provide a safety function" and identify two subtasks that present opportunities for varying depth of reasoning. These subtasks are then analyzed. The final section presents a summary and our conclusions.

OPERATIONAL DEFINITION OF KNOWLEDGE DEPTH

We need a simple relation that will distinguish the depth of models of expertise for a given reasoning task. Our focus is on the explicit representation of knowledge in models of expertise, although other notions of depth, which highlight, for example, multiple perspectives on a domain (Davis, 1984) or notions

of causality (Rieger and Grinberg, 1977; Finin and Morris, 1989), are also potentially valuable for this purpose. The following is intended only as an informal, operational definition of the *deeper-than* relation, which leaves terms such as "knowledge" and "model" to intuition.

> *Definition:* Consider two models of expertise M and M′. We will say that M′ is *deeper-than* M if there exists some implicit knowledge in M that is computed explicitly in M′.

The deeper-than relation is defined over an infinite space of models of expertise for a given reasoning task. In cases where a reasoning task is decomposed into isolated subtasks that present opportunities for varying depth, the relation is applied to subtasks rather than to the composite task. For example, consider a task t that may be naturally decomposed into subtasks $t1$ and $t2$. We address the relative depth of models for these subtasks rather than for t, for if we build a model X and a deeper model X′ for $t1$ and build a model Y and a deeper model Y′ for $t2$, then the composite models for t consisting of {X, Y′} and {X′, Y} are not strictly ordered by deeper-than. This occurs in intelligent safety systems, as described in the next section.

INTELLIGENT SAFETY SYSTEMS AND KNOWLEDGE DEPTH

There has been great interest in intelligent systems that represent and reason about physical devices (Bobrow, 1985). One line of research concerns the development of facilities that provide advice or take direct action in response to system disturbances (Underwood, 1982; Nelson, 1982; Ennis et al., 1986). Of these, we focus on systems that provide *safety functions* in physical systems. Providing a safety function involves executing plans to circumvent potential crises in physical system environments.

In nuclear power operations, this encompasses executing "a group of actions that prevent melting of the reactor core or minimize radiation releases to the public" (Corcoran et al., 1981). Although the term "safety function" originated in the context of nuclear facility management, we can identify applications of the same idea in other domains, including preventing a chemical reactor in a process plant from catching fire and preventing the depletion of operating system queue space in a large computer installation. Providing safety functions in such process environments may be considered an expert-level task, and we will refer to a system that employs models of expertise for providing safety functions as an *intelligent safety system* (ISS).

Generally speaking, an ISS receives a description of the state of the system being controlled (the *target system*) as input and provides a plan of action for

circumventing a crisis as output. The work of an ISS may be naturally decomposed as follows:

Monitoring: An intelligent agent (operator or system) must maintain a model of the target system by periodically obtaining the values of target system state variables. This is necessary in order to detect problem conditions and to ascertain the status of target system components that may be employed in managing those conditions.

Plan determination: An intelligent agent must determine possible plans for managing the current situation. In general, the agent determines a set of alternative plans, which, depending on the domain, may be a complete set or only a set of the most plausible plans.

Plan evaluation: An intelligent agent must evaluate these alternatives plans to select the "best" one. In some domains, this may involve simulating the alternatives to ascertain their outcomes.

Plan execution: An intelligent agent must execute the chosen plan.

Our first task in examining the relative merits of models that vary in depth for a given reasoning task is to identify opportunities for varying depth (in the sense of the preceding section) that provide some potential advantages (in the sense of the first section). For ISSs, monitoring and execution are relatively straightforward operations, but plan determination and plan evaluation may be accomplished in a number of ways that vary in depth of reasoning. The next step in the analysis involves defining and evaluating progressively deeper models for performing each of these subtasks.

REASONING DEPTH IN PLAN DETERMINATION

We examine four progressively deeper models that may be used to determine plans to prevent a crisis: invoking hard-coded plans, determining plans based on hard-coded paths of components, generating plans based on system structure, and generating plans based on system structure and component behavior.

Invoking Hard-Coded Plans

The first model of expertise we consider is the shallowest—hard-coding plans for preventing a potential crisis under various conditions. This model is conveniently implemented in formalisms that encode situation-action pairs such as production rules. As an example, consider JESQ (Klein, 1985), one of several rule-based systems that constitute YES/MVS (Ennis et al., 1986), an expert system for managing large computer installations. JESQ's task is to maintain a


```
Allow-Large-Jobs:
  Priority = 5
  IF remaining-queue-space = low
     printer-setting = only-small-jobs
     small-jobs-left = few
     large-jobs-waiting
  THEN submit command to allow large jobs
```

FIGURE 1. Example of a simple JESQ rule.

"comfortable" level of unused space on an operating system queue, that is, to provide the safety function "prevent queue space depletion." The antecedents of JESQ's rules describe the states under which the hard-coded plans in their consequents should be performed. For example, the rule in Fig. 1 encodes the plan to enable a printer to print large jobs so that queue space may be freed. In effect, it enables a path of data flow from the queue to the printer.

These plans, which specify the movement of data from the queue to other components (e.g., tape drives, printers), are based on the structure of the underlying computer system being modeled, but this structure is only implicitly represented in JESQ. As such, JESQ suffers from some of the disadvantages of shallow models. For example, the configuration of the computer system may be changed, requiring modifications to this and other rules, but there is no systematic way of identifying such modifications. In addition, JESQ's rules may omit reference to conditions in the computer system which do not usually occur but which occasionally render encoded plans unsuccessful. Another limitation of the system is that explanations can offer little more than a presentation of the conditions under which plans are applicable. Finally, JESQ can handle only precisely the state conditions that have been anticipated.

According to our definition, a deeper model would explicitly represent the potential paths of data flow, although deeper does not necessarily imply better. For example, if configuration changes are unlikely or if the encoded preconditions are appropriate most of the time and are of manageable volume, the benefits of a deeper model of plan determination expertise might not justify the cost of its construction or the overhead of its execution.

Determining Plans from Hard-Coded Paths

The next model to be considered involves explicitly representing sets of components that may be employed to provide a safety function. This approach is taken in REACTOR (Nelson, 1982), which provides safety functions in a nuclear reactor facility.

Potential paths of components that can be used to cool the reactor core are encoded in a *response tree* as shown in Fig. 2. Each path in the tree contains an instance of each of the functional components required to provide the safety function (e.g., water source, heat sink). Part of REACTOR's mission is to select, in real time, a path made up of components that are correctly operating.

The primary advantage of this approach is its robustness. Not all potential combinations of component failures need be explicitly anticipated, since these are coordinated by the response tree structure and associated logic. Another advantage is that a change to the configuration is more easily mapped into the representation, since paths of components are explicitly represented. Still implicit, however, is the configuration description from which response trees are constructed. Since the response tree is hard-coded, only the potential paths of components explicitly identified in advance are candidates for selection, and configuration changes require that the resulting new paths be identified by a knowledge engineer. A still deeper model would reason directly from a schematic to *generate* the potential paths. However, for applications in which the number of potential paths is manageable and the structure of the target system is relatively stable, we might not be inclined to consider a deeper model.

Generating Plans from System Structure

The next model that we consider involves explicitly representing the configuration of the target system and using this description to generate plans for circumventing a potential crisis. As an example, we again refer to JESQ's domain as cast in the structural representation of Fig. 3. Using this model, plan determination adopts the form of searching a graph in which nodes represent components and edges represent their interconnections. The search always begins at node QUEUE and terminates at node USER, and each such path through the graph represents a candidate path through which data sets may flow in order to clear the QUEUE.

This representation has the advantage that a change in the configuration may be directly mapped into a change in the representation, and the knowledge about computing plans for moving data sets (searching the graph) remains unchanged. It is also more portable than the other representations, requiring only a configuration description for any particular installation. This model is deeper than the rules' hard-coded plans, which only implicitly represent the queue-clearing paths. This model is also deeper than the response trees of the preceding section, because we have hard-coded the mechanism by which paths are generated rather than the paths themselves.

The queue space domain permits the convenience of uniformly treating each represented device in the system, because each device is capable of accepting and storing data. In domains encompassing devices that do not exhibit this behavioral homogeneity, we would require a still deeper model in order to generate plans. Specifically, we would need to reason explicitly about the behavior of components, since not all components would play the same functional role in providing a safety function. A similar point is made by Ginsberg (1984) regarding diagnostic systems.

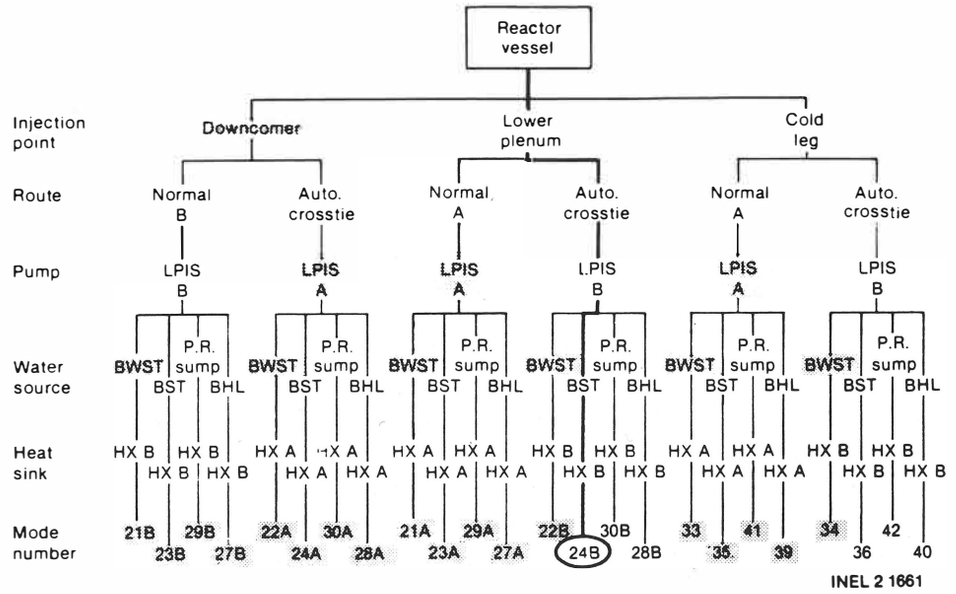

FIGURE 2. Response tree. (From Nelson, 1982.)

Generating Plans from System Structure and Behavior

The deepest model of plan determination we consider is based on an explicit representation of target system structure and component behavior. This approach

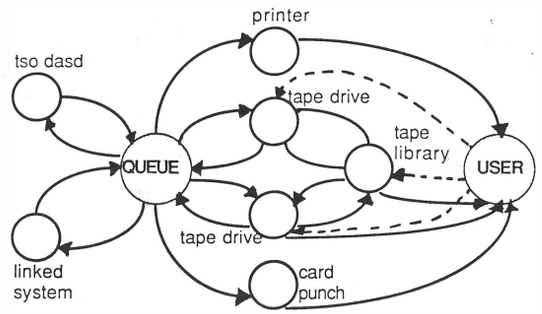

FIGURE 3. Structural model of computer installation.

is taken in several systems that perform other reasoning tasks such as simulation (de Kleer and Brown, 1984), troubleshooting (Davis, 1984), and verification (Barrow, 1984).

As an example of reasoning from structure and behavior to provide a safety function, consider the following hypothetical ISS. Using a spatial representation as in Fig. 4 (Stephanopoulous, 1984) and descriptions of component behaviors, the ISS generates alternative plans for keeping the reactor (RXR1) from catching fire when components fail. For example, if the valve (V3) that regulates the coolant flowing to the reactor jacket is stuck closed, the ISS searches for a compensating action to lower the temperature of the reactor. The search proceeds both forward and backward from the reactor to yield the following alternative plans: close V4, close V5, request A to stop feed stream, request A to lower feed stream temperature, request cool temperature material from the heating system, and request the heating system to stop. For example, the ISS identifies the action "close V4" as follows. The feed stream input to RXR1 must be such that the temperature of RXR1 is normal. Since V3 is stuck closed, the ISS must reduce either the temperature or the flow rate of the feed stream. Searching backward from RXR1, the ISS examines the behavioral model of valve V4, noting that in state CLOSED the flow rate from the valve is zero. The ISS searches a list of potential actions to find that action "close valve" cause V4 to enter state CLOSED. Thus, the plan "close V4" is a candidate for execution to be assessed by the plan evaluation process.

These components (heat exchangers, reactor, valves) are not behaviorally homogeneous, so reasoning about how to maintain a safe temperature in the

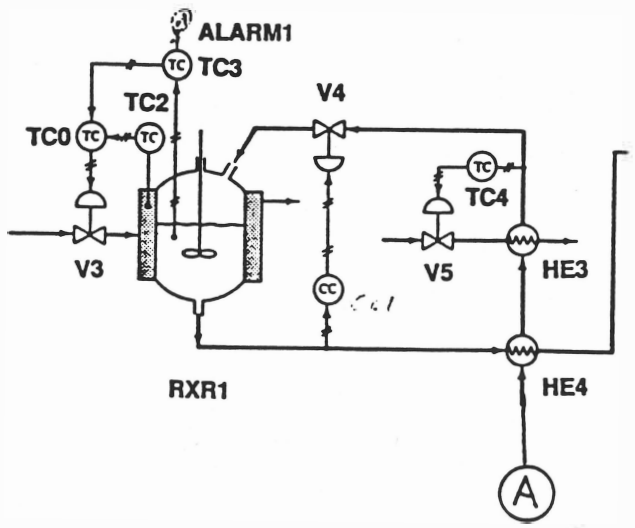

FIGURE 4. Chemical reactor subsystem. (From Stephanopoulous, 1984.)

reactor must encompass consideration of the individual behaviors of the components that affect it, as well as their position in the system structure. This model is deeper than the solely structural model of the preceding section in that the behavior of components is explicitly represented and reasoned about.

VARYING DEPTH OF REASONING IN PLAN EVALUATION

Given a set of alternative plans for providing a given safety function in a particular situation, plan evaluation involves selecting the "best" one. We discuss three significant levels of depth for evaluating alternative plans: hard-coded evaluation (explicit priorities), evaluation using utility theory and hard-coded decision attributes (computation of priorities), and evaluation using utility theory and decision attributes that are themselves computed from a structural and behavioral model of the target system (computation of priorities and of underlying attributes).

Evaluations Encoded as Priorities

The shallowest representation of plan evaluation that we consider is the hard-coded numeric priority. This is a commonly employed approach; for example, a priority is associated with each rule in JESQ and with each path in the response tree in REACTOR to provide for choosing the best when more than one are applicable.

In rule-based systems like JESQ, priorities specify relative preferences to conflict resolution. The antecedent of a rule determines the *eligibility* of the plan in its consequent, and the rule's associated priority indicates its *desirability* relative to other plans. Note that all objectives underlying the desirability of a plan (e.g., minimizing cost, maximizing convenience, maximizing the satisfaction of system users, maximizing speed) are implicitly represented in the priority, and this gives rise to several difficulties.

First, as the rule base grows it becomes difficult to predict the consequences of adding new rules to the knowledge base. In effect, the knowledge engineer must understand the basis for the priorities of all existing rules in order to assign a new priority to a new rule. Second, the meaning of a priority is completely opaque, so there exists no basis for justifying a priority in an explanation. Finally, a system that uses this shallow model of choice will lack robustness. Because great importance is placed on a single heuristically justified symbol, the result of changing the priority of a single rule can significantly alter the overall behavior of an ISS.

This model of plan evaluation suffices when the number of plans is small and the relative desirability of each plan is obvious. For applications not hav-

ing these characteristics, deeper models that explicitly represent the factors underlying priority selection may be useful for automatically justifying a choice among competing plans in an explanation and for facilitating the incremental modification of the knowledge base.

Utility Theory with Encoded Objective Values

The next model to be examined involves explicitly representing the attributes and objectives underlying priority selection in the framework of utility theory. Multiattribute utility theory (Keeney and Raiffa, 1976; Hansen, 1983) is of particular interest in the domain of intelligent safety systems, where multiple, often mutually competitive objectives drive choices between competing plans. Under this approach, we regard each plan's characteristics with respect to each of the objectives that underlie plan evaluation as arguments to a utility function that computes the priority for each plan. The utility function itself abstractly captures the relationships between objectives that underlie the choice of plan and can be thought of as encoding the plant's operational policy.

Employing an explicit model of choice based on utility theory has several advantages over implicit models such as hard-coded priorities. First, since the underlying objectives of plan evaluation are explicitly represented, a basis is provided for generating explanations regarding choices among competing plans. In addition, adding plans to the knowledge base is simplified because the knowledge engineer need only score a new plan with respect to the defined objectives.

For example, consider the hypothetical rules for the JESQ domain given in Figs. 5 and 6. Each rule consists of an *event* specification that describes the relationships among target system state variable values and determines the eligibility of the plans. The plan itself is laid out in the *response* portion of the rule. Finally, the *response attributes* portion of the rule describes it in terms of underlying objectives.

Either rule may be appropriate, depending on the status of the target system at the time of instantiation. If, say, printer2 is not free but tape1 is, then the *copy* rule of Fig. 5 would be executed. Alternatively, if printer2 is free but tape1 is not, then the *expensive printing* rule shown in Fig. 6 would be employed. If neither device is free we are out of options.* But what if both tape1 and printer2 are free? Which plan is better? We essentially need to choose between the two heuristics according to the installation's operational policy, which enables one to determine the preferred plan using the response attributes (Klein, 1989).

*In an actual implementation, we would also encode heuristics that involve deliberate action to free them.


```
ALTERNATIVE copy:
    EVENT:
       queue space is low
       dataset
             name = ds1
             size = 1,000,000 lines
             destination = printer1
       printer status
             name = printer1
             status = broken
       tape drive status
             name = tape1
             status = free
    RESPONSE:
       copy ds1 to tape1
       move tape on tape1 to tape library
       move tape from tape library to tape1
       copy ds1 from tape1 to queue
       print ds1 on printer1
       move ds1 from printer1 to user bins
    RESPONSE ATTRIBUTES:
       excess operator time   = 10.0
       excess turnaround time = 34.2
       difference in form     = 1.0
       excess cost            = 1.0
       queue clearing time    = 15.1
END ALTERNATIVE
```

FIGURE 5. Rule for copying a dataset to a tape.

Where such scores for the response attributes of plans are reasonably easy to formulate, this second level of depth suffices for plan evaluation. However, in applications where scores for objectives encompass consideration of the behaviors of large sets of components, formulating scores may be difficult, encouraging a greater level of depth. Specifically, we may wish to reason about (or compute) the scores for objectives rather than assign them. In applications that employ a model of plan determination that actually generates plans, a deeper model of plan evaluation will be required, for there will be no way to assign scores in advance to plans that are constructed during problem solving.

Utility Theory with Computed Objective Values

The deepest model of plan evaluation that we consider for ISSs involves computing the scores for underlying objectives that are input to the utility

```
ALTERNATIVE expensive printing:
    EVENT:
       queue space is low
       dataset
             name = ds1
             size = 1,000,000 lines
             destination = printer1
       printer status
             name = printer1
             status = broken
       printer status
             name = printer2
             status = free
    RESPONSE:
       route ds1 from printer1 to printer2
       print ds1 on printer2
       move ds1 from printer2 to user bins
    RESPONSE ATTRIBUTES:
       excess operator time   = 0.1
       excess turnaround time = 0.0
       difference in form     = 1.0
       excess cost            = 100.00
       queue clearing time    = 25.0
END ALTERNATIVE
```

FIGURE 6. Rule for printing a dataset on a more expensive printer.

function. This may be accomplished by formulating another set of (hard-coded) data from which objective values may be computed for each plan. In applications that employ a model of plan determination that actually generates plans, we would use the same representation of the target system to support both plan determination and evaluation. For example, consider augmenting the spatial description of the computer system of Fig. 3 with component descriptions (e.g., processing time per line of data) pertaining to the objectives mentioned earlier (e.g., maximize user satisfaction). A plan (path of devices) can be evaluated with respect to turnaround time (one aspect of user satisfaction) by summing the processing times of the processors that lie along the generated path. Other objectives (e.g., speed of action, work for the operator) would be similarly computed.

This model of evaluation is deeper than that of the preceding section because, rather than encoding the values for objectives, we encode functions for computing them. One advantage of this method is that we need only supply local device-dependent data for each represented device in order to compute the desirability of actions, rather than making subjective judgments about the desirability of predefined paths.

SUMMARY AND CONCLUSIONS

We have characterized the depth of models of expertise in terms of knowledge they explicitly represent and reason about. For the reasoning task "provide a safety function," we identified two subtasks that provide opportunities for building progressively deeper models of knowledge, described some particular models for performing each subtask, and reviewed their relative merits for some particular domains.

If knowledge engineering is to become more of a discipline than an art, we will need to develop some guidelines that more precisely characterize "depth of knowledge" and its implications for intelligent system construction, performance, and maintenance. Ultimately, the guidelines would provide a basis for selecting among models of varying depth based on general domain characteristics. We believe that such guidelines should be developed by circumscribing isolated reasoning tasks and analyzing the relative merits of models of varying depth inspired by various domains, and we have sketched one such analysis in this paper. Future work will employ alternative definitions of depth in the analysis and analyze other reasoning tasks to provide the data on which the mentioned guidelines may be based.

REFERENCES

Barrow, H. 1984. VERIFY: A program for proving correctness of digital hardware designs. *Artif Intell* 24(1–3):437–491.

Bobrow, D., ed. 1985. *Qualitative Reasoning about Physical Systems.* Amsterdam: Elsevier.

Chandrasekaran, B., and Mittal, S. 1983. Deep versus compiled knowledge approaches to diagnostic problem solving. *Int J Man-Machine Studies* 10(5):425–436.

Corcoran, W., Finnicum, D., Hubbard, R., III, Musick, C., and Walzer, P. 1981. Nuclear power-plant safety functions. *Nucl Safety* 22(2).

Davis, R. 1984. Diagnostic reasoning based on structure and behavior. *Artif Intell* 24(1–3):347–410.

de Kleer, J., and Brown, J. S. 1984. A qualitative physics based on confluences. *Artif Intell* 24(1–3):7–83.

Ennis, R., Griesmer, J., Hong, S., Karnaugh, M., Kastner, J., Klein, D., Milliken, K., Schor, M., and Van Woerkom, H. 1986. A continuous realtime expert system for computer operations. *IBM J Res Dev* 30(1):14–28.

Finin, T., and Morris, G. 1989. Abductive reasoning in multiple fault diagnosis. *Artif Intell Rev* (in press).

Fink, P. 1985. Control and integration of diverse knowledge in a diagnostic expert system. *Proc. IJCAI* 426–431.

Ginsberg, A. 1984. Localization Problems and Expert Systems. CBM-TR-139, Rutgers University.

Hansen, P., ed. 1983. *Essays and Surveys on Multiple Criteria Decision Making.* New York: Springer-Verlag.

Hart, P. 1982. Directions for AI in the eighties. *SIGART Newslett,* no. 79, January, 11–16.

Keeney, R., and Raiffa, H. 1976. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* New York: Wiley.

Klein, D. 1985. An Expert Systems Approach to Realtime, Active Management of a Target Resource. MS-CIS-85-40, University of Pennsylvania.

Klein, D. 1989. Ph.D. Dissertation, University of Pennsylvania, forthcoming.

Klein, D., and Finin, T. 1987. What's in a deep model? A characterization of knowledge depth in intelligent safety systems. *Proc. IJCAI-87,* 559–562.

Koton, P. 1985. Empirical and model-based reasoning in expert systems. *Proc. IJCAI,* 297–299.
Nelson, W. 1982. REACTOR: An expert system for diagnosis and treatment of nuclear reactor accidents. *Proc. AAAI,* 296–301.
Rieger, C., and Grinberg, M. 1977. The declarative representation and procedural simulation of causality in physical mechanisms. *Proc. IJCAI,* 117–122.
Stephanopoulous, G. 1984. *Chemical Process Control.* Englewood Cliffs, N.J.: Prentice-Hall.
Underwood, W. 1982. A CSA model-based nuclear power plant consultant. *Proc. AAAI,* 302–305.