Modeling and Diagnosing Problem-Solving System Behavior

EVA HUDLICKÁ AND VICTOR LESSER

Abstract—**A new component of a problem-solving system, called the diagnosis module (DM), that enables the system to reason about its own behavior is described. The aim of the diagnosis is to identify inappropriate control parameter settings or faulty hardware components as the causes of observed misbehavior. The problem-solving system being diagnosed is a distributed interpretation system, the distributed vehicle monitoring testbed (DVMT), which is based on a blackboard problem-solving architecture. The diagnosis module uses a causal model of the expected behavior of the DVMT to guide the diagnosis. Causal-model-based diagnosis is not new in AI. What is different is the application of this technique to the diagnosis of problem-solving system behavior. Problem-solving systems are characterized by the availability of the intermediate problem-solving state, the large amounts of data to process, and in some cases, the lack of absolute standards for behavior. New diagnostic techniques that exploit the availability of the intermediate problem-solving state and address the combinatorial problem arising from the large amount of data to analyze are described. A technique has also been developed, called comparative reasoning, for dealing with cases where no absolute standard for correct behavior is available. In such cases the diagnosis system selects its own "correct behavior criteria" from objects within the problem-solving system which did achieve some desired situation. The diagnosis module for the DVMT has been implemented and successfully identifies faults.**

I. Introduction

THE COMPLEXITY of man-made systems is rapidly increasing to the point where it is becoming difficult for us to understand and maintain the systems we build. Artificial intelligence (AI) problem-solving systems are particularly susceptible to this information overload problem due to their often *ad hoc* design, large knowledge bases, and decentralized control mechanisms. This has recently resulted in a trend toward more autonomous systems: systems that can explain their behavior, aid the developers with debugging, and monitor and adapt their behavior to changing requirements. Central to all these functions is the ability of the problem-solving system to reason about its own behavior.

In this paper we describe a component of a problem-solving system, the diagnosis module (DM), that reasons about a problem-solving system's behavior to diagnose the faults responsible for inappropriate system behavior. The DM has been implemented and successfully diagnoses faults in a distributed problem-solving system, the distributed vehicle monitoring testbed (DVMT) [11]. The faults diagnosed may be either hardware failures (e.g., a failed sensor) or inappropriate parameter settings, which we call *problem-solving control errors* (e.g., the confidence factor assigned to a sensor's output). The DM consists of about 5000 lines of Lisp code and runs on a VAX under the VMS operating system.

The Diagnosis Module

By way of an example, let us motivate the use of a diagnosis module in a problem-solving system. Suppose that a problem-solving system fails to generate the desired result. In our case the DVMT-distributed interpretation system fails to track a vehicle in some part of the sensed environment. Knowing the general characteristics of the desired result, the DM traces back through the history of problem solving guided by the model of correct processing. The DM determines what intermediate results would have had to be produced to achieve the desired results. In this way the cause of the failure to generate a desired result can be traced back to the lack of low-level data; this problem could then possibly be traced further to a failed sensor or an incorrect setting of the control parameter that specifies the confidence factor associated with the sensor's output. In the latter case diagnosis involves understanding that data from the sensor were available but not processed because they were below an acceptable confidence threshold.

Characteristics of Diagnosis of Problem-Solving System Behavior

Diagnosis is not a new problem for AI. Many systems exist for medical diagnosis [14], [16], diagnosing digital circuits [4], [6], [9], electrical devices [12], and large systems such as nuclear reactors [13]. We have found that diagnosis of problem-solving system behavior is different from the techniques used in these domains. While some diagnostic techniques are generally applicable across all domains,[1]

Manuscript received January 15, 1986; revised November 8, 1986. This work was supported in part by the National Science Foundation under Grants NSF DCR-8500332 and NSF DCR-8318776 and by the Defense Advanced Research Projects Agency (DOD) monitored by the Office of Naval Research under Contract N00014-79-C-0439, P00009.

E. Hudlická was with the Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003. She is now with the Advanced Systems and Tools Group at DEC, HL2-3/C10, 77 Reed Road, Hudson, MA 01749.

V. Lesser is with the Computer and Information Science Department, University of Massachusetts, Amherst, MA 01003.

IEEE Log Number 8714383.

[1] For example, given a symptom, go back through the events that led up to it until the cause is found.

0018-9472/87/0500-0407$01.00

aspects of problem-solving system behavior require new diagnostic techniques.

Problem-solving systems are characterized by:

- complete knowledge of the internal system structure;
- availability of the intermediate problem-solving state;
- large amount of data to process during diagnosis;
- in many cases, lack of absolute standards for correct behavior.

Since the structure of the system is known, we can use a causal model of the system in which problem-solving behavior is modeled as a series of states. These states are linked by causal relationships to represent the sequence of events required for a desired result to be produced. Diagnosis then consists of determining why some expected state was not reached by exploring the appropriate part of the causal model of the system.

Since the internal system state is available (by directly examining the system data structures), we need not address the problem of determining the internal state from the inputs and outputs.[2] However, we introduce a correspondingly difficult combinatorially explosive problem: the states of the causal model must be mapped onto the record of system behavior. In many cases a number of possible intermediate results from this record could be used in diagnosis. For example, the problem-solving system may have partially explored a number of alternative paths in attempting to generate a solution. All of these are stored in the system as part of the intermediate state record. In a multilevel blackboard system such as the DVMT, the intermediate state includes the many possible hypotheses on the blackboard which *could* have been on the path to a solution. The crucial question is, does diagnosis require an exhaustive analysis of all the search paths explored by problem-solving system search or is there a way for diagnosis to limit its analysis?

Much of the work reported here was devoted to developing techniques for avoiding the potential combinatorial explosion of diagnostic paths to analyze by choosing which particular piece of data to use in diagnosis and how to group related diagnostic paths. For example, the formalism for modeling the problem-solving system allows the representation of a *class of objects* so that during diagnosis, the DM can reason about classes of situations rather than individual cases.

Dealing with Lack of Absolute Standards for Behavior

Causal model diagnostic techniques work as long as a model of the expected behavior is available. Such a model requires the existence of absolute criteria for system behavior. In most cases we can provide such criteria when dealing with problem-solving systems. For example, an expected sequence of events in the DVMT system behavior is the creation of a hypothesis, followed by the creation of a goal, and then followed by the scheduling of a knowledge source. Cases exist, however, where no absolute criteria exist and a fixed model for correct system behavior cannot be constructed *a priori*. Instead, we need to compare the behavior of the faulty object to a similar object that seems to behave correctly. In understanding why these objects differ, we often uncover a fault. This lack of absolute criteria for system behavior led to the development of a diagnostic technique we call *comparative reasoning* (CR). When using CR, the diagnosis module examines similar cases within the system and from these chooses a standard with which to compare the suspect situation; this comparison is accomplished using a simple form of qualitative reasoning [3]. A model is thus dynamically constructed where both causal analysis and qualitative reasoning are used to analyze the factors responsible for the suspected situation.

Let us look at a simple example of this type of reasoning in the DVMT, which is an agenda-based problem-solving system. The agenda contains a list of knowledge source (KS) processes that could be executed. They are ordered by their rating, which is a function of a number of parameters. Suppose that the DM traces some symptom to a KS that did not execute because its rating was too low. The next step is to discover which of the parameters influencing the rating is responsible for the low overall rating of the KS. However, no absolute standards exist for any of these parameters. The comparative reasoning analysis involves first selecting a similar KS; this can be accomplished by choosing one that is at the top of the agenda and thus likely to execute or one that has already executed. The next step involves the pairwise comparison of the parameters influencing the KS rating for the low-rated KS and the high-rated KS. Suppose the low-rated KS that did not execute (i.e., the problem KS) has two parameters, A-problem KS and B-problem KS, and the high-rated KS that is used for comparison (i.e., the model KS) has the same parameters, A-model KS and B-model KS. Further, suppose the comparison of parameter values reveals A-problem KS = A-model KS, but B-problem KS $\ll$ B-model KS. The diagnostic module's analysis can then conclude that the B parameter was responsible for the low rating of the problem KS. This intermediate result in the diagnosis can be traced further; for example, the B parameter could be low because it was based on the belief associated with the KS's input data. This low belief could then be traced further to the low setting of the parameter controlling the confidence level that the system associates with the sensor generating the data.

Comparative reasoning brings up many interesting problems. The choice of a good object to use as a model for the object of interest is a nontrivial task, as is the matching of the parallel states in the two instantiated models.

[2] Many other diagnostic systems have dealt with the type of reasoning necessary given a blackbox view of the system (Genesereth's DART [6] and Davis's digital circuit analyzer [4] systems deal mainly with the problems associated with this view), and our modeling formalism supports this type of reasoning. What the availability of the intermediate states allows us to do is to get beyond these reasoning mechanisms and explore other interesting problems associated with diagnosing the behavior of complex systems.

The rest of the paper is organized as follows. Section II briefly describes the distributed vehicle monitoring testbed (DVMT). Sections III and IV discuss the modeling formalism and provide a more detailed description of the diagnostic reasoning techniques. An indepth description of a diagnostic session is developed through an example in Section V. The example illustrates the use of the comparative reasoning technique. Section VI discusses the methods we have developed for reducing the combinatorics explosion resulting from the large amounts of data to diagnose. Section VII summarizes the work and outlines some directions for future research.

II. Context: The DVMT

The problem-solving system we model and diagnose is called the DVMT, a distributed problem-solving system in which a number of processors cooperate to interpret acoustic signals. The goal of the system is to construct a high-level map of vehicle movement in the sensed environment. Raw data are sensed at discrete time locations at the signal level. The final answer is a pattern track describing the path of vehicles, moving as a unit in some fixed pattern formation. To derive the final pattern track from the individual signal locations, the data undergo two types of transformation: the individual locations must be aggregated to form longer tracks, and both the tracks and the locations must be driven up several levels of abstraction, from the signal level, through the group and vehicle levels, up to the pattern level (see Fig. 1).

Each processor in the DVMT system is based on an extended Hearsay-II architecture where data-directed and goal-directed control are integrated [1]. The problem-solving cycle at each processor begins with the creation of a hypothesis that represents the position of a vehicle. Hypotheses then generate goals that represent predictions about how these hypotheses can be extended by incorporating more of the sensed data. Finally, a hypothesis together with a goal triggers the scheduling of a knowledge source (knowledge source instantiation) whose execution will satisfy the goal by producing a more encompassing hypothesis (one which includes more information about the vehicle motion). This cycle begins with the input data and repeats until a complete map of the environment is generated. Fig. 2 illustrates the processing structure at each node.

III. Structure of the System Behavior Model

This section describes the modeling formalism used to represent the possible behaviors of the DVMT system. By representing the internal structure of the DVMT (i.e., the causal relationships among DVMT events), we generate the system behavior model (SBM) that supports not only diagnosis, but also simulation of the system behavior. Unlike some other causal models, such as CASNET [16], which represent causal relationships among pathological states, the SBM represents the normal system behavior. The errors are represented as deviations from the expected

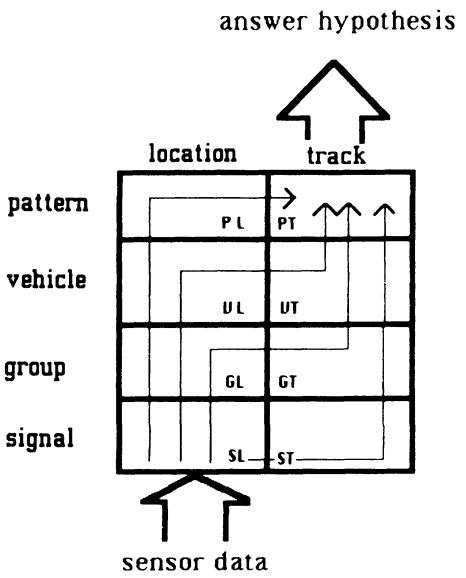

Fig. 1. Levels of abstraction in DVMT data transformations. Data blackboard has eight levels of abstraction. Input data, acoustic signals representing vehicle positions at discrete time intervals, come in at signal level (sl). Final answer, at pattern track level, is integrated picture of "raw" sl data representing how vehicles move through environment.

situations that were not achieved by the system. The model can thus reason both about the causal sequences of expected events (thereby simulating the correct system behavior) and about the sequences of abnormal events (thereby diagnosing faulty system behavior). The work of Genesereth [6] and Davis [4] is similar in that it uses the violated expectations approach to diagnosis.

The system behavior in the DVMT is modeled by a set of causally related states corresponding to a series of events in the system. Each event results in the creation of an object (e.g., hypothesis, goal, or knowledge source instantiation) or the modification of the attributes of some existing object. *The states in the model represent the results of such events in the DVMT system.* Depending on what we want to model, a state may represent simply whether some event has occurred, or it may represent some finer aspect of the event's outcome.[3]

The SBM formalism consists of three major components: *hierarchical state transition diagrams* which represent the possible system behaviors at various levels of detail; *abstracted objects* which represent individual objects (i.e., data structures) or classes of objects in the DVMT system; and *constraint expressions* among the different attributes of the abstracted objects which represent the relationships among the objects. We can view the model as two parallel networks (see Fig. 4).[4] At the higher level is the state transition diagram, consisting of states and directed state transition arcs. At the lower level is the network consisting of the abstracted objects whose attri-

[3] There are two types of states in the model. *Predicate states*, which represent whether an event has occurred or not, and *relationship states*, which represent the relationship among two objects in the DVMT. The relationship states are used in comparative reasoning.

[4] Fig. 3 contains the legend for the figures in this paper.

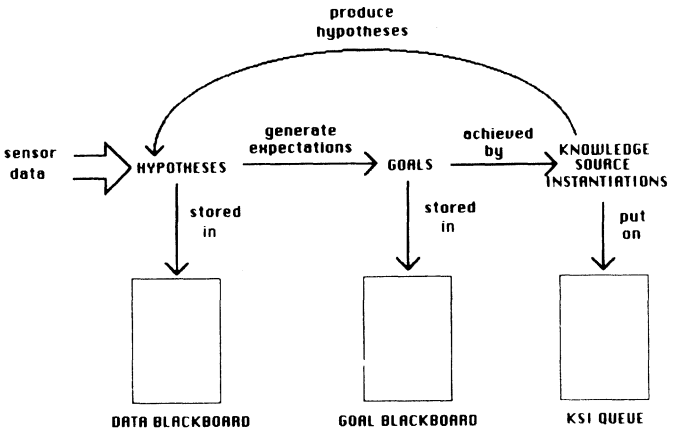

Fig. 2. Structure of processing at each node in DVMT system. DVMT begins its interpretation task with arrival of sensed data. All data are represented by hypotheses and stored on data blackboard. Arrival of hypothesis stimulates creation of goal, which represents prediction of how hypothesis might be extended in future. Hypothesis together with goal stimulate instantiation of knowledge source (KSI). Each such instantiation is rated and if rating is high enough, KSI is inserted onto scheduling queue. At beginning of each system cycle highest rated KSI executes and produces additional hypotheses. Cycle repeats until final answer is derived or until there are no more data to process.

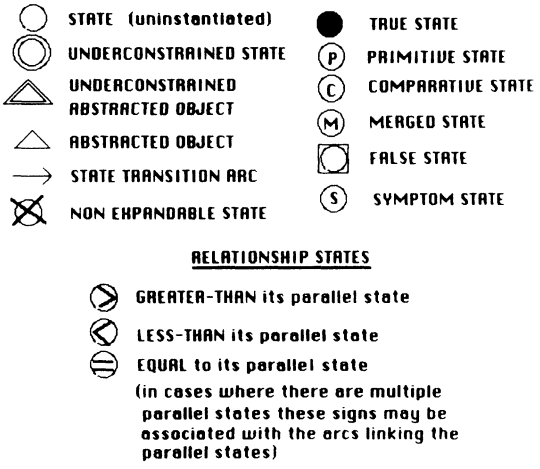

Fig. 3. Legend for figures in paper.

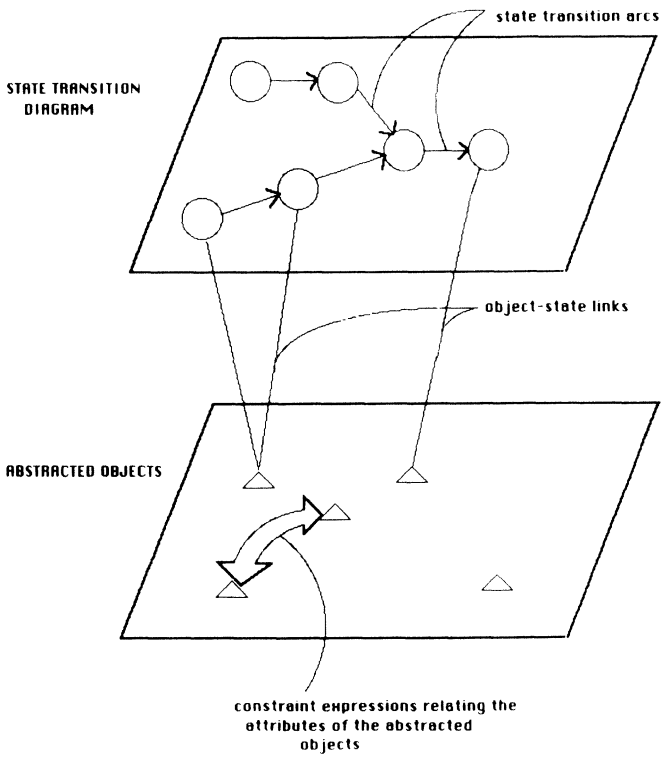

Fig. 4. High-level view of modeling formalism. SBM modeling formalism consists of three major components: state transition diagram clusters representing expected sequences of events in DVMT, abstracted objects representing DVMT objects such as hypotheses or goals, and constraint expressions representing relationships among attributes of neighboring abstracted objects. SBM can thus be viewed as two parallel networks: one containing state transition diagrams, other containing abstracted objects and constraint expressions.

butes are linked by the constraint expressions that capture the relationships among the object attributes. The constraint expressions relating the attributes of two abstracted objects are shown in Fig. 5. The two networks are connected by state–object links.

States are linked to other states to form an AND/OR graph. If an event is influenced independently by a number of preceding events, then the states representing these events will be ORed. That is, any one of the preceding events determines the outcome of the event in the same manner. If the outcome of an event is influenced by a number of preceding events acting together, then the states representing these events will be ANDed. Fig. 6 shows a part of the system model.

The states are linked to the abstracted objects which describe characteristics of objects in the problem-solving system record. Abstract objects are represented as frames. If all attribute slots of a frame have fixed values, the abstracted object corresponds to a specific instance of an object in the system. When some abstract object attributes are not specified precisely, the abstracted object is *underconstrained* and represents a whole class of objects. For example, an underconstrained hypothesis object could have a list of levels in its level attribute and thus represent the entire class of hypotheses at any of those blackboard levels. If objects exist in the DVMT that match the characteristics of an abstracted object, then the desired event that is specified by the state/object has occurred. The objects are represented as separate entities from the states for efficiency reasons to avoid the duplicate representation of similar sets of object attributes since several states may refer to the same object.

The state transition diagram representing the system behavior is organized into small clusters for manageability (see Fig. 7). These clusters are then organized into a hierarchy corresponding to increasingly detailed views of the system. Thus a high-level cluster represents selected events as contiguous states while a more detailed cluster represents other events which occur in between these states. Such a hierarchical representation allows reasoning at different levels of abstraction. This is useful during diagnosis because it allows the system to focus quickly on the problem by postponing a more detailed analysis until it is

This attribute points to objects that exist in the DVMT. If no such objects exist, it is false.

```
((vmt-ids (f     find-track-hyps
   (path self time-location-list)
   (path self event-class)
   (path self node)
   pt))
```

This attribute represents the path of the vehicle.

```
(time-location-list
     (((pt message-ob) (path x1 tll/trl))
     ((pt vt-hyp-ob) (path x1 time-location-list))
     ((pt pl-hyp-ob) (path x1 time-location-list))
```

This attribute represents a creation of shorter pt segments that could produce the desired segment. The function *create-trace-segments* looks for existing shorter segments and then chooses the longest non-overlapping ones for instantiation.

```
     ((pt pt-hyp-ob) (f create-track-segments
      (path x1 time-location-list)
      (path x1 event-class)
      pt
      (path x1 node)))))
```

This attribute represents the specific type of signal. The function phd:higher-level-event-classes determines the event classes for the pattern level from the vehicle level according to the system signal grammar.

```
(event-class   (((pt message-ob) (path x1 event-classes))
                ((pt vt-hyp-ob)
  (f     higher-level- event-classes
          vt
                  (path x1 event-class)))
 ((pt pt-hyp-ob) (path x1 event-class))
 ((pt pl-hyp-ob) (path x1 event-class))))
(level          pt)
(node  (path x1 node)))
```

Fig. 5. Constraint expressions among abstracted objects. Constraint expressions linking abstracted object attributes allow DM to determine attribute values of one object based on attribute values of any of its neighboring objects. Figure shows relationship among attributes of object representing pattern track hypothesis and its neighboring objects: shorter pattern tracks (pt-hyp-ob), pattern location hypotheses (pl-hyp-ob), vehicle track hypotheses (vt-hyp-ob), and received pattern tracks (message-ob). First part of each constraint expression specifies context: current state and neighbor whose values are to be used. For example, (pt message-ob) means that current state is pt and object whose values are to be used is neighboring message object. Second part of expression specifies which of object's attribute values should be used. For example, (path $x1$ tll/trl) specifies tll/trl (time-location-list/time-region-list) attribute of neighboring object (represented by variable $x1$ which always refers to neighbor object that was instantiated most recently and whose values are to be used).

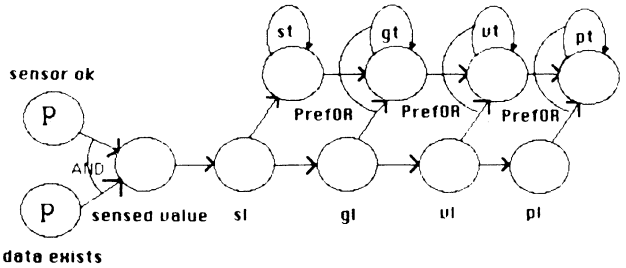

Fig. 6. Answer derivation model. Model cluster represents data transformation DVMT is expected to perform; that is, to produce pattern track (pt) hypotheses from incoming sensor data at signal level (sl). Arrival of initial data depends on sensor functioning (SENSOR-OK state) and data's existence (DATA-EXISTS state). SENSED-VALUE state represents separate sensed signal for each sensor.

necessary. For example, consider the case when the DM tries to determine why some hypothesis was not constructed. Rather than looking for the knowledge source instantiation that could have produced that kind of hypothesis, the DM first looks for the necessary supporting data, and only if these data exist does it investigate the knowledge source instantiations to see whether they were scheduled and if so, why they did not run. This means that the diagnosis is first done using the answer derivation cluster and only later using the KSI scheduling cluster[5] (see Fig. 7). A subset of the states, designated as primitive, represents reportable faults during diagnosis.

The SBM thus represents a generalized description of selected aspects of the DVMT system behavior. Fig. 7 shows the set of model clusters we have constructed.

IV. Use of SBM in Reasoning about DVMT Behavior

Reasoning about DVMT behavior consists of instantiating the part of the SBM that represents the system behavior relevant to the situation being analyzed. The aim of all the different reasoning strategies is to explore the causes, or the effects, of an initial situation provided as input to the DM.[6] This situation is represented by an instantiated

[5] The KSI scheduling cluster represents the events that occur in between each pair of states in the answer derivation cluster.

[6] Currently, this is done by hand. In a fully fault-tolerant system the input would come from a detection component.

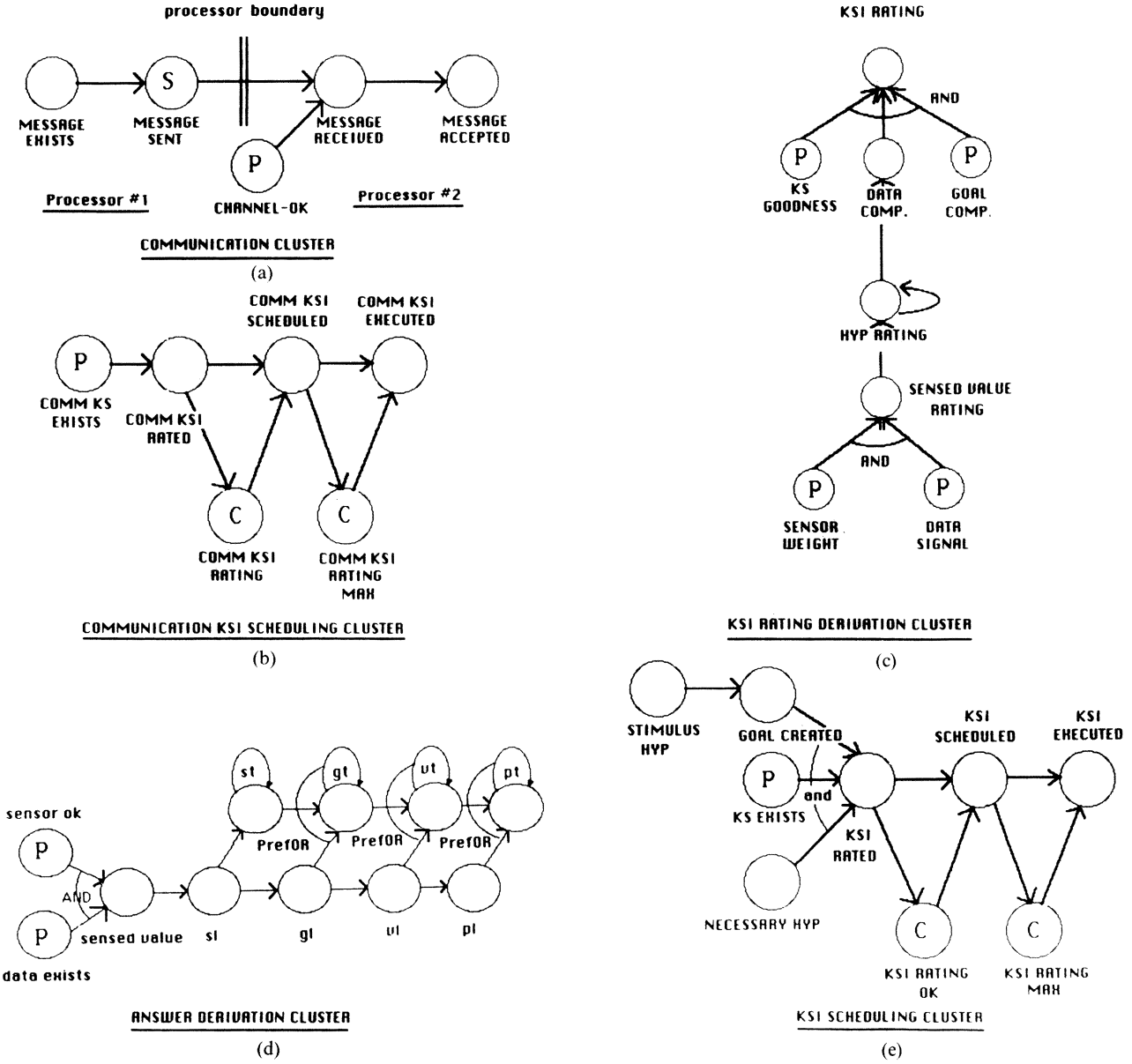

Fig. 7. Model clusters representing DVMT behavior. Figure shows diagrams of all five model clusters used in diagnosis. At highest level are answer derivation cluster and communication cluster. KSI scheduling and Comm KSI scheduling clusters represent events that take place in between each pair of states at higher level models. KSI rating derivation cluster represents additional knowledge about value relationships among rating components of various objects. (a) Communication cluster. (b) Communication KSI scheduling cluster. (c) KSI rating derivation cluster. (d) Answer derivation cluster. (e) KSI scheduling cluster.

state and its abstracted object, that is, a state and object whose attributes have been evaluated (see Fig. 8). The DM propagates these known values through the SBM using the constraint expressions among the abstracted objects and thereby instantiates a sequence of states causally related to the initial situation. We call such a sequence a *causal pathway*.

When the initial situation represents some desirable event in the DVMT that never occurred, we call it a *symptom*. A symptom represents an object that was never created by the DVMT, usually a hypothesis.[7] Upon receiving a symtpom, the DM traces back through the SBM to find out at which point the DVMT stopped working "correctly." This is done by comparing the behavior necessary for the desired situation to occur, as represented by the instantiated model, with what actually did occur in the problem-solving system, as determined from the DVMT data structures. The aim is to construct a path from the symptom state to some false primitive states which caused it and thus explain, in terms of these primitive causes, why the DVMT system did not behave as expected. This type of diagnostic reasoning thus consists of backward chaining through the SBM. Since it constructs a causal pathway linking the initial symptom to the faults that caused it, we call this type of reasoning *backward causal tracing* (BCT). Fig. 9 shows a BCT-constructed causal pathway.

The BCT search stops when all possible pathways relevant to the situation being analyzed have been explored.

[7]A symptom could also represent a class of objects, such as all hypotheses at some blackboard level. This would be done using under-constrained objects.

State MESSAGE-SENT0111

f-n	(p:or (message-received0110))	; front neighbor
b-n	(p:or (message-exists0217)))	; back neighbor
value	F	; state is false
object-ptrs	message-ob0071	; abstracted object

Abstracted Object MESSAGE-OB0071

vmt-ids	F
from-node	2
message-type	hyp-ob
tll/trl	((5 (14 10)) (6 (16 12)) (7 (18 14)) (8 (20 16)))
event-classes	1
level	vt
node	2
to-node	3

Fig. 8. Symptom represented by instantiated state and abstracted object. Situation in DVMT is represented by instantiated state and its associated abstracted object. Figure shows instantiated MESSAGE-SENT state and its object MESSAGE-OBJECT. Object represents hypothesis at vehicle track (vt) level that was to be sent from Node 2 to Node 3. That this hypothesis was never sent is represented by value attribute of the state, which is f (false). Only subset of attributes is shown.

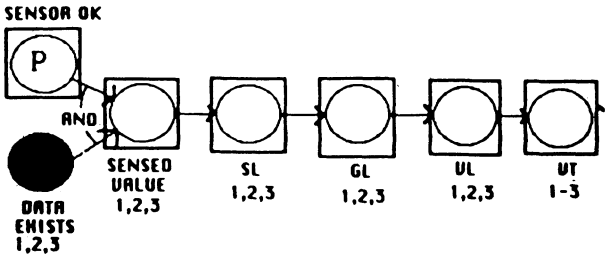

Fig. 9. BCT-constructed causal pathways. Instantiated answer derivation cluster where BCT traced lack of vehicle track (vt) 1–3 hypothesis to failed sensor, represented by false SENSOR-OK state. Instantiated model shows intermediate states that form causal pathway from false symptom state VT to primitive false state SENSOR-OK. Intermediate states are VL state representing three necessary locations and, similarly, GL and SL states, and SENSED-VALUE state, representing signals sensed by individual sensors (in this case only one sensor exists and therefore only one sensed value).

For example, to determine why some hypothesis was not constructed, the DM must examine all possible ways in which it could have been generated: that is, via several pathways within a node, from locally available data, or from data received from other nodes. Within the instantiated model the analysis is done exhaustively by a depth-first search. An exhaustive search is feasible here because the search space has already been reduced by the methods to be discussed in Section VI: for example, grouping together objects that behave similarly, using underconstrained objects to reason about a class of objects rather than the individual cases, and using existing data to constrain the search.

In addition to symptoms, initial situations may represent arbitrary events in the DVMT whose effect on the DVMT behavior needs to be simulated. This is the case, for example, when the DM needs to see what effects some identified fault, such as a faulty parameter setting or a failed hardware component, has on the DVMT system. This type of simulation thus consists of forward chaining through the SBM. Here the DM constructs a causal pathway which links the initial situation to all situations caused by it. We therefore call this type of reasoning *forward causal tracing* (FCT). FCT uses underconstrained objects to reason about the class of problems caused by the fault, rather than just the individual cases.

Both BCT and FCT are more complex than simple backward and forward chaining because the model is hierarchical and the DM must decide when to change the level of resolution, for instance, when to reason at different levels of abstraction and when to reason about classes of objects. In addition to BCT and FCT, we have also found the need for a new type of diagnostic reasoning we call *comparative reasoning*, which will be described in detail through an example in the following section.

V. Diagnostic Session Using Comparative Reasoning

This section describes a DM diagnostic session of a failure scenario where the use of comparative reasoning is necessary to handle situations in which no absolute criteria for correctness exist. CR works by selecting a situation from the DVMT which can be used as a model with which to compare a problematic situation. CR then compares these two situations in the DVMT system and tries to explain why they are different. This is done by systematically tracing the development of both situations and comparing them at each step. This type of reasoning is necessary because we cannot always understand the system behavior by looking at an isolated object in the system and comparing that object to some fixed standard, as is done with backward causal reasoning.

The following example illustrates the use of comparative reasoning to track the low rating of a knowledge source instantiation (KSI) to a low-rated hypothesis at an early point in the data transformation. The diagnosis begins with a missing high-level hypothesis at the pattern track (pt) level.[8] The diagnosis module reconstructs, based on a causal model of processing in the DVMT, the internal events and intermediate results that would be required to generate the desired data. As a result of this backward trace, it discovers that the desired hypothesis was not derived because lower level location hypotheses were never produced. Specifically, while hypotheses did exist at the group location (gl) level, they were never driven up to the next level, the vehicle location (vl). The diagnosis module further determines that this was due to the fact that the

[8] We will be following the convention of representing both symptoms and faults (i.e., any undesirable situations) by false states in the case of predicate states and by the qualitative relationships lower-than or greater-than in the case of relationship states. For example, a lack of some hypothesis at the vehicle track (vt) level will thus be represented by a false state VT, with its associated abstracted object representing the specific vt hypothesis.

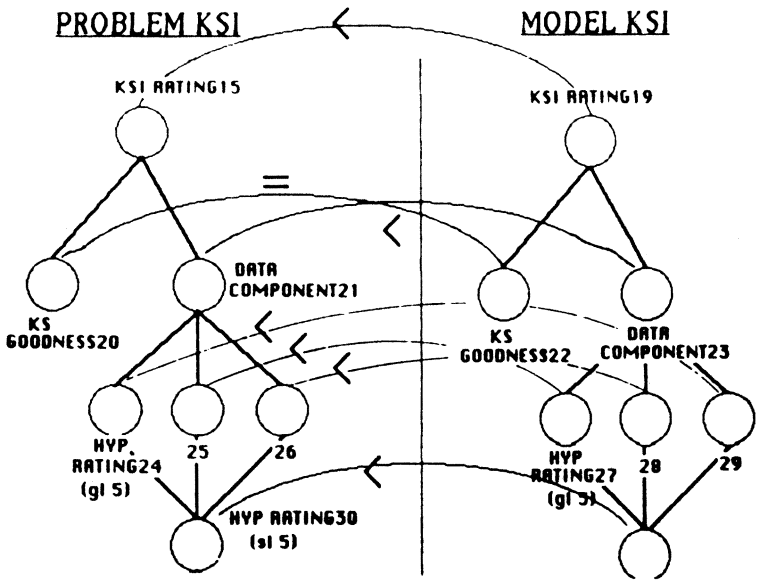

Fig. 10. Instantiated SBM for fault scenario 2. Instantiated SBM for diagnosis of low KSI rating by comparative reasoning. Comparative reasoning works by instantiating two copies of model cluster, in this case KSI rating derivation cluster, and comparing "problem object rating" with "model object rating." "Model object" is chosen by diagnosis module based on selection criteria contained in SBM. Here low KSI-RATING15 is traced to low HYP-RATING30 of sl hypothesis.

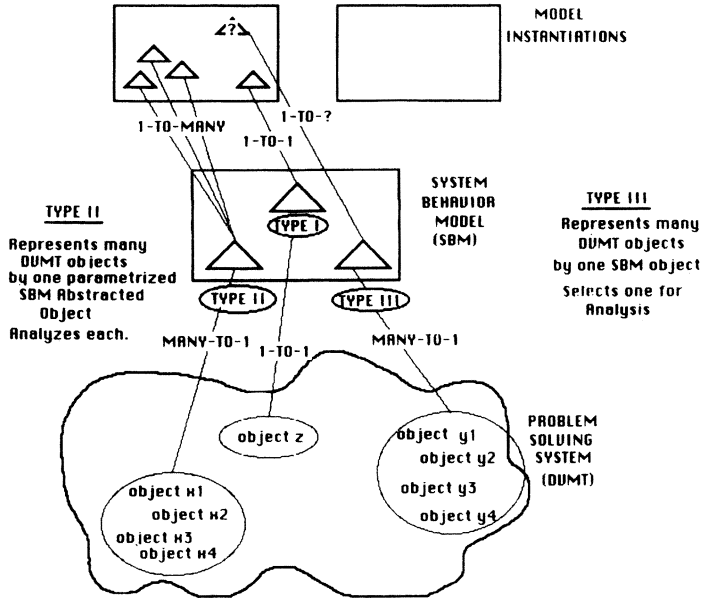

Fig. 11. Types of abstractions used in model construction and reasoning. Figure shows types of abstractions necessary to represent complex system. Bottom part of figure represents DVMT. Middle part represents uninstantiated SBM, and upper part represents two instantiations of SBM.

KSI that would have created the desired vl hypotheses never executed because its rating was too low. Recall that it is always the highest rated KSI on the scheduling queue that executes. It is thus possible for a KSI with a low rating to remain on the queue for a long time. The diagnosis up to this point is done via backward causal tracing. The low-rated KSI is represented by the state KSI RATING15 in Fig. 10.[9]

The point of the diagnosis here is to determine what caused the KSI rating to be low. The DM now switches to comparative reasoning and to a cluster representing the derivation of the KSI rating, the KSI rating cluster. Let us call the low-rated KSI the *problem KSI* and the maximally rated KSI on the queue (with the appropriate characteristics) the *model KSI*. Before CR can continue, a model object must be found for the low-rated KSI. Such an object must be of the same type of the problem object and it must, of course, be rated higher than the problem object. In this case we are looking for a "successful" KSI (i.e., a high-rated KSI which is about to run) that takes a hypothesis at the gl level and produces a corresponding hypothesis at the vl level. The model KSI is represented by the state KSI-RATING19 in Fig. 10. The DM has a model of how a KSI rating is derived from its components: a KS

[9] This is a relationship state whose value is "lower-than."

parameter called KS-goodness, the goal that is to be satisfied by the hypotheses produced by the KSI, and the data the KSI is to use.

In this case, CR examines the factors influencing the KSI rating for both the problem KSI and a model KSI selected from the queue. It notes that both the KS-goodness parameters and the goal components are identical in both the problem and the model KSI's but that the data component rating is lower for the low-rated KSI. This then is identified as the cause of the overall low rating of the problem KSI. The next step is to trace back through the derivation of this low hypothesis rating and determine why *it* is rated low. Again, the highly rated hypothesis from the model KSI is used as a standard with which to compare the low-rated hypothesis. This continues through several levels of abstraction until either a primitive node in the model is reached which is responsible for the low rating (in this case, a low sensor weight or low-rated data are identified) or if the search is unsuccessful, until the causal pathway can no longer be extended.

Notice that there are now two parallel instantiations of the KSI rating cluster: one for the problem KSI and one for the model KSI. The two clusters will be instantiated one state at a time and compared in an attempt to find an explanation for the low rating of the problem KSI. The search through the model (i.e., which neighbors will be expanded next) is now determined by the type of relationship found among the problem state and the model state (i.e., $<$, $=$, or $>$), rather than the state value (true or false), as was the case with predicate states. CR continues to expand back neighbors as long as they can explain the current state's relative value with respect to the parallel state. A state's relative value is explained by its predecessor states" values if they have the same relationship. For example, a $<$ state is explained by its preceding $<$ states, but not by $>$ or by $=$ states.

The value for state KSI-RATING15 is, of course, $<$ (this is guaranteed by the process selecting the model object; if no appropriate model exists, no object will be instantiated, the problem state will not have a parallel state, and diagnosis will stop). To understand why KSI-RATING15 is low,[10] the DM expands its back neighbors to see if any of them are abnormally low. Since a KSI rating is a function of the KS goodness (a parameter which determines the quality of a knowledge source) and the data component (the ratings of the hypotheses the KSI is working with), the back neighbors of the state KSI-RATING are KS-GOODNESS and DATA-COMPONENT. These are instantiated, and their values are determined from the values of their corresponding objects in the DVMT system. The resulting states are KS-GOODNESS20 and DATA-COMPONENT21.

Similarly, the back neighbors of the model KSI rating state KSI-RATING19 must be expanded so that the comparisons can continue. This expansion produces the states KS-GOODNESS22 and DATA-COMPONENT23. Before the values of these states can be determined, the parallel states must be matched. (Recall that the value of relationship states is determined by comparing the ratings of the problem and model objects.) The problem state KS-GOODNESS20 is matched with the model state KS-GOODNESS22, and the problem state DATA-COMPONENT21 with the model state DATA-COMPONENT23. The relationship of $=$ is found for the KS-goodness states, because the KS-goodness values are identical. The relationship of the data-component states is $<$, because the value of the data-component rating of the problem KSI is lower than the value of the data-component rating of the model KSI.

The next step is to select a subset of the expanded back neighbors to expand further. As in backward causal tracing, we want to continue expanding only those states that explain the current problem. The current problem is a low KSI rating, and we have determined that one of the components influencing this rating is normal (KS goodness) and one is below normal (data component), where "normal" means "same as the parallel state." Clearly, the normal value could not have caused the KSI rating to be low. The KS-GOODNESS20 state is, therefore, a dead end as far as the diagnosis is concerned because this state is not causally related to the KSI-RATING15 state. The low DATA-COMPONENT21 state, however, is responsible for the low KSI-RATING15, and we therefore follow this state backward by expanding its back neighbors.

The value of the data component of a KSI is a function of the data (i.e., the stimulus and the necessary hypotheses). In this case there were three gl hypotheses, one for each event class, whose rating determined the rating of the data component. (Knowledge sources that transform lower level hypotheses into higher level ones often combine several low-level hypotheses of different event classes into one higher level one.) The back neighbor of data-component, hyp-rating, is therefore instantiated into three states, one for each of the three hypotheses of the problem KSI. These states are HYP-RATING24, HYP-RATING25, and HYP-RATING26. Their associated abstracted objects are objects representing group location hypotheses, GL-HYP-OB's.

The state matching is more difficult here than before because there are three parallel states to choose from on the model side; each problem hyp-rating state has to select one of the model hyp-rating states. In this case a heuristic is used to select the appropriate model state: the DM looks for a model state that minimizes the difference between the ratings of the two objects while maintaining the constraint that the model rating must be higher than the problem rating. (This is discussed in more detail in [8].) In the current example, this difference minimization results in the following state pairs: HYP-RATING24 and HYP-RATING29, HYP-RATING25 and HYP-RATING27, and HYP-RATING26 and HYP-RATING28. The values of these states are $<$ since all the hypotheses ratings are lower than the model hypotheses ratings. Diagnosis, therefore, continues with the expansion of the back neighbors

[10] Low here really means "lower than the model object" since there is no such thing as absolutely low or high.

of these states. We begin with the state HYP-RATING24 and expand its back neighbors. The back neighbor of this state is another hyp-rating state, representing the rating of the hypothesis at the sl level from which the gl hypothesis referred to by the state HYP-RATING24 was derived. The resulting state is HYP-RATING30. We also expand the back neighbor of the parallel state which results in the instantiation of the state HYP-RATING31. The state matching is trivial here since there is only one model state to choose from. The two states are matched, the value of state HYP-RATING30 is determined to be lower than its parallel state HYP-RATING31, and diagnosis continues by expanding the back neighbors of this state. Thus the low rating of a KSI at the vl level has been traced to a low-rated hypothesis at the sl level. We will end the diagnosis here, although it normally continues all the way to the sensor-weight and data-signal states, which represent the primitive causes that are ultimately responsible for the hypotheses ratings. See [8] where this example is discussed in more detail.

VI. Reducing the Combinatorial Problems in Diagnosis

In a typical run the DVMT creates hundreds of objects in each of its processing nodes. To diagnose a given situation, a subset of these objects has to be represented in the instantiated SBM; the model then has to be searched in an attempt to find the causes for the situation. This section describes the methods for dealing with the potential combinatorial explosion resulting from the large amount of data stored in the problem-solving system record (see Fig. 11). The following is a list of these methods:

1) parametrizing a group of objects to represent the entire group by one parametrized abstracted object (occurs during model construction);
2) parametrizing groups of states to represent them by one state in the model (occurs during model construction);
3) allowing the existing data to constrain the search during diagnosis (occurs during model instantiation);
4) selecting a representative from a group of related objects and reasoning about it (occurs during model instantiation);
5) grouping similar objects together and reasoning about them as a group to reduce the search (occurs during model instantiation);
6) abstracting the common characteristics of a group of objects to represent and reason about the group by one abstracted object, usually an underconstrained object (occurs during model instantiation).

This rest of this section motivates and describes methods 3–6. The first two are discussed in detail in [8, ch. 6].

Constraining the Search by Existing Data

In some cases the number of possible ways that a DVMT object could have been derived is too large to be able to explore each possible derivation path. One way of constraining the search without eliminating the diagnosis of a possible fault is to use the existing data generated by the DVMT to rule out certain paths. This situation occurs, for example, in track hypotheses elongation, where a number of shorter track segments hypotheses or individual location hypotheses are combined to form a longer track hypothesis.

Suppose we are trying to analyze why a track consisting of eight locations was not created. We could consider all the possible combinations of shorter track segments and locations and analyze why they were not created. In these cases not only would the combinatorics be prohibitive, but it would not even be useful to explore all the possible derivation paths since the system would not explore all of them either, being constrained by the data it has. We can reduce the number of pathways to explore by allowing the existing data to constrain the search during diagnosis. Rather than exploring all the possible ways of deriving some longer track, we consider only the track segments that have already been derived by the system and analyze why they were not extended further.

Selecting a Representative Object from a Class of Objects

Another use of abstraction involves the selection of a representative object from a group of objects and analyzing its behavior rather than the behavior of each of the individual objects. For this strategy to be effective we must guarantee that the set of faults diagnosed when analyzing the representative object is the same as the set of faults diagnosed if each of the objects was analyzed separately. As for the foregoing method, track elongation best illustrates the use of this technique. To create a longer track hypothesis, the DVMT system will aggregate shorter track segment and location hypotheses. Therefore, at any time a number of shorter track segments will exist. In this case we choose the longest segment and analyze why it was not extended further. This does not reduce the number of faults we can identify because the undiagnosed shorter track segments fall into one of two categories. In one case the shorter undiagnosed segments were later extended into a longer segment and no fault exists. In the other case, the reason they were not extended is the same as the reason the longest track of the group was not extended. This fault will be identified by analyzing why the longest track segment was not extended, thus bypassing identical (redundant) diagnosis for the shorter track segments.

Grouping Together Similarly Behaving Objects

In cases where a single object in the model is expanded into a number of objects in the instantiated model, it may be possible to group some of these objects together and diagnose each group as a unit. This is more efficient than creating a separate state for each of the objects and then repeating identical diagnostic paths with each of the states. In the initial stages of this project every object created in the instantiated SBM had its corresponding state created and attached. During diagnosis then, each of these state–object pairs would be processed. This led to a combi-

natorics problem, even in relatively simple cases. Consider the diagnosis of the following situation. A pattern track hypothesis ranging from time 1 through time 8 is missing. To diagnose it, the system instantiates the SBM and traces the problem to the lack of the necessary data for this hypothesis. Because the hypothesis has eight time locations, eight locations are missing. Thus there are eight pathways to follow during diagnosis. Notice also that they all reduce to the same problem; missing data. It would be much more efficient if we could recognize that all these objects behave similarly and, therefore, require the same diagnosis and can be grouped and diagnosed together. In other words, all the objects behaving similarly can be grouped under one state. This state is then the only one that needs to be followed during diagnosis because the predicate or value it represents is the same for all its associated objects. There is, therefore, only one diagnostic path to follow through the model.

Currently, the grouping is performed by applying a function to some combination of the state or object attributes. The result of this function determines the number of equivalence classes for the objects. A separate state is then created for each of those equivalence classes, and all the objects in that class are attached to that state. This object grouping greatly reduces the number of paths that need to be examined during diagnosis without sacrificing the completeness of the diagnosis. As soon as the object behavior changes and requires a different diagnostic pathway, the system creates a separate state for it as specified in the grouping criteria. An example of a criterion for grouping objects is the existence of a corresponding object in the DVMT. All the abstracted objects which do not have a corresponding object in the DVMT are grouped together under one state (which is false), and all those that do are grouped under another (which is true).

Underconstrained Objects

Another way of using abstraction when reasoning about the system is to group together a number of objects and represent the whole class as one abstracted object in the instantiated model. This is done by underconstraining some of the attribute values of the abstracted object. Underconstrained objects are useful when it is known that a group of objects will behave identically and we can, therefore, save time by reasoning about the group as a whole. A situation where this is useful is the simulation of the effects of an identified fault. Suppose the system has identified a bad sensor in the DVMT system. It would be useful to propagate the effects of this sensor forward through the model and thereby not only explain any pending symptoms, which were caused by the same fault, but also account for future symptoms. In this case the simulation involves reasoning about the class of hypotheses generated from data in the area covered by the failed sensor. Clearly, it is more efficient to represent this whole class as one object rather than reasoning about all the possible hypotheses. Examples where this occurs are found during the forward simulation of an identified fault.

The following example illustrates the use of underconstrained objects to represent and reason about a class of objects. This capability is then applied to simulating the effects of an identified fault on system behavior. Suppose a problem-solving system is configured such that four communicating nodes exist, each working on its part of the overall problem. Each node has different parameters that control various aspects of its processing. For example, a set of parameters controls the internode communication (i.e., who should talk to whom and about what). If these parameters are set incorrectly, then the entire system will fail in its problem solving because the different parts of the overall solution cannot be integrated.

The lack of an overall solution will result in each node having a set of symptoms to diagnose relating to missing solution parts due to lack of communication. The DM will begin with one of these symptoms, for example, the lack of a specific piece of data, and will trace it to a failure: an incorrect communication parameter setting. Once this fault is identified, its effects will be propagated through the system. As an example, consider what would happen if two nodes were expected to exchange data in a certain area, but the communication parameters were wrong and no messages were sent. This would be discovered by the DM when it is determined that a particular piece of data is missing from one of the nodes. When the fault is identified, it is generalized to reflect that no data in the entire affected region will be communicated. In this way all symptoms representing missing data in that region will be accounted for when the effects of the fault are simulated.

To summarize, due to the complexity of a problem-solving system, as compared to other systems to which causal model diagnosis has been applied, we have utilized a number of techniques to handle the combinatorial problems. In this section we have highlighted the types of abstractions necessary to make problem-solving system modeling and diagnosis feasible.

VII. Summary and Future Research

This paper discussed our work in the area of *problem-solving system diagnosis*. Our approach integrates a number of known techniques (diagnosis, simulation, qualitative reasoning, and constraint networks) and describes two new ones (comparative reasoning and the use of underconstrained abstracted objects) in an attempt to solve the problems encountered in representing and reasoning about problem-solving system behavior.

We have implemented a component of a problem-solving system, the diagnosis module, that diagnoses faults in the problem-solving system behavior by using a causal model of the system. The faults can be either hardware failures or inappropriate control parameter settings, which we call *problem-solving control errors*.

Our approach to diagnosis has been determined by the following characteristics of the DVMT problem-solving system.

- The system maintains a number of data structures (blackboards, queues) that contain its recent history. We exploit this availability of the system's intermediate states in constructing an instantiation of the system model to represent how a particular situation was reached.
- For many events in the DVMT system no absolute criteria exist for behavior. This means that in many cases we cannot determine whether an event is appropriate simply by comparing it to some fixed ideal event. We must instead take a more global view and examine the event's relationship with other events in the system. The lack of absolute standards for system behavior necessitates a new type of diagnostic reasoning we call comparative reasoning.
- Because of the complexity of the DVMT system, we could not represent every aspect of the system in the model. The problem of devising a concise representation of a large set of possible system behaviors led to various types of abstractions, both in the model construction and in the use of the instantiated model. These abstractions allow us to represent and reason about classes of objects rather than individual cases.

To perform diagnosis with these constraints, we have built upon a number of techniques developed elsewhere. We use causal networks similar to CASNET [16], except that we model correct rather than faulty behavior. We have also added the explicit representation of objects in the diagnosed system, in addition to representing the sequence of expected states. The mapping between these abstracted objects in the model and the data structures in the system is not always straightforward; we cannot just represent each object in the problem-solving system by a corresponding abstracted object due to the large number of system objects. We have developed techniques for reducing these combinatorial problems. We have parametrized abstracted objects and underconstrained objects to represent classes of situations in the problem-solving system. We have also developed techniques for dynamically selecting a specific system object to represent, which will capture the necessary characteristics of the situation that needs to be diagnosed. We have exploited the techniques developed by Genesereth [6] and Davis [5] in forward and backward reasoning from first principles. Again, however, we have had to extend these techniques to handle the increased complexity of a problem-solving system as compared with simple digital circuits. Our use of fault simulation represents synthesis of both techniques to understand what symptoms can be explained as a result of an identified fault. Finally, the work on comparative reasoning represents a new technique built on the basic backward/forward causal reasoning and qualitative analysis technique developed elsewhere. The diagnosis module is currently being used to help explain the DVMT system behavior, and we are planning to integrate it into the DVMT to provide more sophisticated metalevel control [7].

Although this work was done with a specific system in mind, we believe that the modeling formalism as well as the diagnostic techniques we have developed are equally applicable to other systems, problem-solving or otherwise, that have knowledge of the internal system structure and access to the system's history. The techniques we developed help alleviate problems associated with the diagnosis of a large number of cases (underconstrained objects) and make possible the diagnosis of some cases where no absolute criteria exist for system behavior (comparative reasoning). Although we have only used underconstrained objects in simulating the effects of a fault, it is extendable to generalizing over a group of symptoms and thus combining many diagnostic paths into one. Comparative reasoning was used to compare why two ratings of knowledge sources differed. It could easily be extended to comparing other quantities, such as the length of tracks or length of derivation paths. We see many possibilities for further research in this area based on our initial experience.

1) The abstracted objects could be extended to represent object components and composite objects and reasoning techniques could be devised for these new object types. Such an object hierarchy could be used to represent both low-level domain and system knowledge and high-level expectations about the system behavior. We have already begun work in this area.

2) The model could be extended to represent not only what the system should do but the assumptions underlying the reasons for doing it. This would permit a much deeper analysis of the system behavior.

3) The issues could be formalized in comparative reasoning, and comparative reasoning extended to be able to compare several system runs with different parameter settings and understand why they differ.

4) Our approach relies on the availability of the intermediate problem-solving states to reconstruct the actual system behavior. In large systems this becomes a problem as the system would quickly become flooded with information. Another area of future research would attempt to develop techniques for reducing the amount of information kept by the system and yet maintain enough so that past behavior can be reconstructed.

The work described here is a first pass at this large problem. Diagnosis is a pervasive activity in problem-solving system development and use; it plays a role in debugging, in metalevel control, and in explaining system behavior. We believe we have demonstrated that causal-model-based diagnosis of problem-solving system behavior is feasible and that while our work deals with a specific problem-solving system, the techniques described here are applicable to other systems.

Acknowledgment

We would like to thank our colleagues Paul Cohen, Susan Conry, and Daniel Corkill, as well as the reviewers, for their careful reading of this article and their thoughtful suggestions for modification.

REFERENCES

[1] D. D. Corkill, V. R. Lesser, and E. Hudlická, "Unifying data-directed and goal-directed control: An example and experiments," in *Proc. 2nd Nat. Conf. Artificial Intelligence*, Aug. 1982, pp. 143–147.

[2] D. D. Corkill, "A framework for organizational self-design in distributed problem-solving networks," Ph.D. dissertation, Dept. Computer and Information Science, Univ. Massachusetts, Amherst, Feb. 1983.

[3] S. E. Cross, "An approach to plan justification using sensitivity analysis," *Sigart*, vol. 93, pp. 48–55, July 1985.

[4] R. Davis *et al.*, "Diagnosis based on descriptions of structure and function," in *Proc. 2nd Nat. Conf. Artificial Intelligence*, Aug. 1982, pp. 137–142.

[5] R. Davis, "Diagnostic reasoning based on structure and behavior," *Artificial Intelligence*, vol. 24, pp. 347–410, 1985.

[6] M. Genesereth, "Diagnosis using hierarchial design models," in *Proc. Nat. Conf. Artificial Intelligence*, Aug. 1982, pp. 278–283.

[7] E. Hudlická and V. Lesser, "Meta-level control through fault detection and diagnosis," in *Proc. Nat. Conf. Artificial Intelligence*, Aug. 1984, pp. 153–161.

[8] E. Hudlická, "Diagnosing problem-solving system behavior," Ph.D. dissertation, Dept. Computer and Information Science, Univ. Massachusetts, Amherst, Feb. 1986.

[9] V. E. Kelly and L. I. Steinberg, "The CRITTER system: Analyzing digital circuits by propagating behaviors and specifications," in *Proc. Nat. Conf. Artificial Intelligence*, 1982, pp. 284–289.

[10] V. R. Lesser and L. D. Erman, "Distributed interpretation: A model and an experiment," *IEEE Trans. Comput.*, Special Issue on Distributed Processing Systems, Vol. C-29, pp. 1144–1162, Dec. 1980.

[11] V. Lesser and D. D. Corkill, "The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks," *AI Mag.*, vol. 4, pp. 15–33, Fall 1983.

[12] D. McDermott and R. Brooks, "ARBY: Diagnosis with shallow causal models," in *Proc. Nat. Conf. Artificial Intelligence*, 1982, pp. 370–372.

[13] W. R. Nelson, "REACTOR: An expert system for diagnosis and treatment of nuclear reactor accidents," in *Proc. Nat. Conf. Artificial Intelligence*, Aug. 1982, pp. 296–301.

[14] R. S. Patil, P. Szolovits, and W. B. Schwartz, "Causal understanding of patient illness in medical diagnosis," in *Proc. 7th Int. Joint Conf. Artificial Intelligence*, vol. 2, 1981, pp. 893–899.

[15] C. Reiger and M. Grinberg, "The declarative representation and procedural simulation of causality in physical mechanisms," in *Proc. 5th Joint Conf. Artificial Intelligence*, vol. 1, Aug. 1977.

[16] S. M. Weiss, C. A. Kulikowski, S. Amarel, and A. Safir, "A model-based method for computer-aided medical decision making," *Artificial Intelligence*, vol. 11, pp. 145–172, 1978.

Eva Hudlická received the B.S. degree in biochemistry from the Virginia Polytechnic Institute and State University, Blacksburg, the M.S. in computer science from The Ohio State University, Columbus, and the Ph.D. degree in computer science from the University of Massachusetts, Amherst, in 1986.

She has recently joined the Advanced Systems and Tools Group at DEC after spending a year as a Visiting Assistant Professor at the University of Massachusetts. Her research interests include knowledge-based diagnosis, causal models, and model-based development environments for AI software.

Dr. Hudlická is a member of ACM and AAAI.

Victor Lesser, for a photograph and biography please see page 379 of this TRANSACTIONS.