META-LEVEL CONTROL THROUGH FAULT DETECTION AND DIAGNOSIS

Eva Hudlicka and Victor R. Lesser
Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts, 01003

ABSTRACT

Control strategies in most complex problem-solving systems, though highly parameterized, are not adaptive to the characteristics of the particular task being solved. If the characteristics of the task are atypical, a fixed control strategy may cause incorrect or inefficient processing. We present an approach for adapting the control strategy by introducing a meta-level control component into the problem-solving architecture. This meta-level control component is based on the paradigm of Fault Detection/Diagnosis. Our presentation will concentrate on modeling the problem-solving system and on the inference techniques necessary to use this model for diagnosis. We feel that meta-level control based on the Fault Detection/Diagnosis paradigm represents a new approach to introducing more sophisticated control into a problem-solving system.

I INTRODUCTION

This paper explores the use of meta-level control in a problem-solving system to adaptively change the system's control parameters in order to make problem solving more robust and efficient. In many complex problem-solving systems the control strategies are highly parameterized. These parameters control decisions such as:

1. what importance to attach to information generated by different sources of knowledge;
2. what type of search to perform (e.g., breadth vs. depth first; data vs. goal directed);
3. what type of predictions to generate from partial results;
4. what criteria to use to judge whether a solution is acceptable.

These parameter settings, which are often determined in an ad hoc manner, are based on typical characteristics of the tasks being posed to the problem-solving system and the characteristics of the problem-solving system itself. Even though such a parameterization makes it relatively easy to change control strategies, the system is rarely allowed to change its own control parameters as the task or system characteristics change during processing. Thus, if the characteristics of a particular task are atypical or the system characteristics* change during execution, the resulting incorrect parameter settings may cause inefficient or incorrect processing.

Our approach to adapting these problem solving control parameters is to introduce a meta-level control component into the problem-solving system architecture, based on an extension of the Fault** Detection/Diagnosis (FDD) paradigm [4, 5] to handle problem-solving control errors resulting from inappropriate parameter settings. The FDD system has three components: the Fault Detection module, the Fault Diagnosis module, and the Strategy Replanning module. See Figure 1 for a diagram of the system architecture. The Fault Detection module monitors the state of problem solving in order to detect when the problem-solving system's behavior deviates from the expected behavior. The criteria for expected behavior are based on standards for acceptable problem solving performance and internal consistency in the problem-solving system data base. Examples of detection criteria are:

1. a large number of highly rated processing goals not being achieved;
2. tasks on the problem solving agenda being too low rated or the agenda being empty;
3. low credibility of intermediate results or contradictory information being generated;
4. results not being produced in a timely fashion or no results being produced for problems where a solution is expected.

If such a situation is encountered by the Fault Detection module, the Fault Diagnosis module is invoked to analyze why the situation occurred. The Diagnosis module, using a detailed model of the problem-solving system and the current state of problem solving, determines which control parameter settings were responsible for reaching the undesirable situation. A Strategy Replanning module is then invoked to adjust the parameters so that appropriate problem solving activities are performed.

This research was sponsored, in part, by the National Science Foundation under Grant MCS-8306327 and by the Defense Advanced Research Projects Agency (DOD), monitored by the Office of Naval Research under Contract NR049-041.

* Previous work has examined this approach in a distributed problem-solving environment where it is likely for processors, communication channels, and sensors to be faulty [9].

** We use the term fault in a very liberal sense to include inappropriate parameter values.

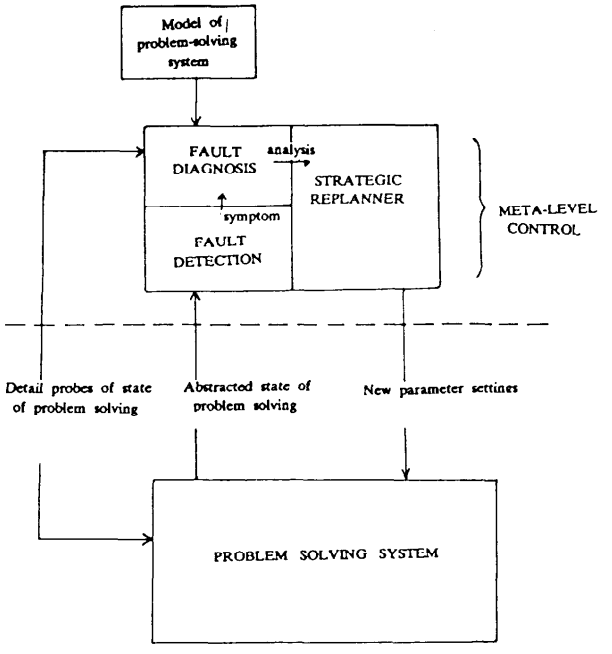


Figure 1: System Architecture.

This approach to meta-level control, which involves adapting the control strategies, is a generalization and extension of earlier work by Hayes-Roth and Lesser on policy knowledge sources for Hearsay-II [8], the Hayes-Roths multi-level control structure for planning [7], and Wilensky's work on meta-level control [13]. It is, however, much different in character and emphasis from the work on meta-level control by Davis [3], Genesereth and Smith [6], and B. Smith [12]. Though the general frameworks they posit for meta-level control can be used to build the type of meta-level control proposed here, their emphasis is different. Their work is oriented more towards how to *layer control knowledge within a single uniform inference framework* to accomplish each control decision rather than the *type of knowledge and inference* required to introspect about the behavior and the performance of the system. It is this latter orientation which will be the focus of the remainder of this paper.

We will illustrate the use of our approach to adaptive control by examining the knowledge and inference structure necessary to implement the Fault Diagnosis module for a problem-solving system based on a goal-directed Hearsay-II architecture, the Vehicle Monitoring Testbed (VMT) [11]. The task of this system is to interpret acoustic signals produced by vehicles moving through a two-dimensional area and generate a map of the environment, indicating what types of vehicles there are and what paths they took. Section II describes how we model the VMT system structure and function. Section III illustrates by way of example how this model is used by the Diagnosis module of the FDD system to diagnose a faulty parameter setting. Section IV describes the status of the system and directions for future research.

II MODELING A PROBLEM-SOLVING SYSTEM

This section describes our model of the Vehicle Monitoring Testbed (VMT) problem-solving system and explains how this model can be used to understand why the system arrived at a particular state. The VMT system derives its results from the input data (see Figure 2a) by incrementally constructing and aggregating intermediate level hypotheses until hypotheses that represent a complete map of the environment are generated. As part of the processing of the system, the creation of an intermediate hypothesis causes the generation of several types of goals. These goals are descriptions of the classes of higher level hypotheses that can potentially be generated given the existence of the newly created hypothesis [2]. Once a goal has been generated, the system attempts to satisfy the goal by scheduling and executing knowledge sources to produce the higher level hypotheses. This is the basic system cycle.

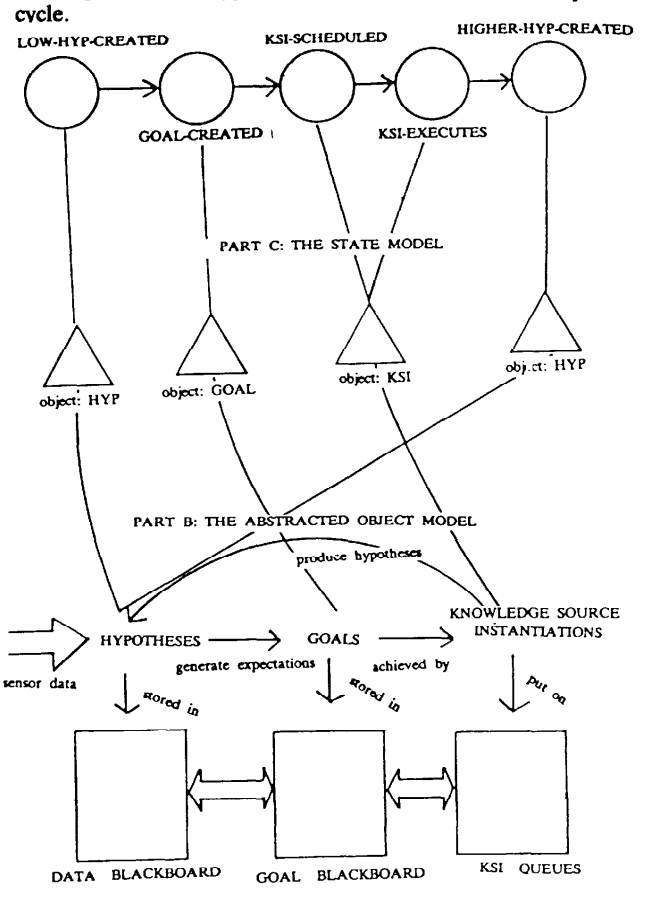


Figure 2: Modeling the VMT Problem-Solving System.

This figure illustrates the state transition/abstracted object model of the VMT system, a high level view of the system structure, and the relationship among them.

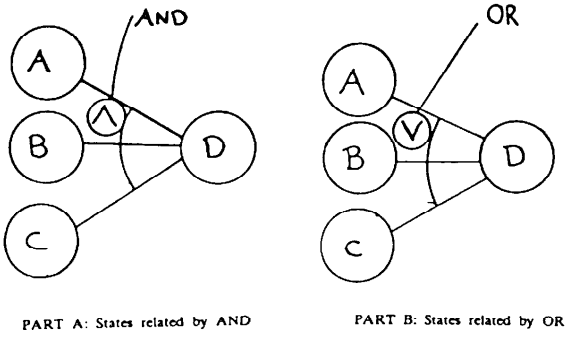
The system behavior thus consists of a series of events. Each event results in the creation of an object (e.g., hypothesis, goal, or knowledge source instantiation) or the modification of the attributes of some existing objects. We can represent the system behavior by specifying either the events or the changes these events cause in the system in terms of their effects on the attributes of the system objects. We chose the latter as the basis for our representation and model the problem-solving system behavior by a state transition diagram (see Figure 2c). Each state represents a specific state of some object in the VMT system in terms of its attribute values. Each state is specified by a schema, which contains *links* to other states in the model (such as states preceeding it and following it), pointers to the descriptions of the system objects the state refers to (these descriptions of the VMT objects are called *abstracted objects*; see Figure 2b), and a *constraint expression* over the abstracted objects' attribute values. This constraint expression is evaluated during diagnosis to determine whether the state has been reached by the VMT system; i.e., whether there exist objects in the problem-solving system whose attribute values satisfy the constraint expression associated with the state.

For example, the process of generating a hypothesis at a higher level of abstraction from one at a lower level of abstraction can be described as follows: the creation of a lower level hypothesis causes the creation of a goal to produce a specific result (i.e., the higher level hypothesis) that incorporates the lower level hypothesis. This causes the scheduling of a knowledge source instantiation (KSI) which later executes and produces the higher level hypothesis. In our model this series of events is represented as the sequence of states: LOW-HYP-CREATED, GOAL-CREATED, KSI-SCHEDULED, KSI-EXECUTES, and HIGHER-HYP-CREATED (see Figure 2). The state transition arcs, which connect the individual states in the model, represent causal relationships among the states. In some cases there may be more than one state transition arc coming in or out of a given state. For example, in Figure 3, states A, B, and C precede state D. The model needs to represent the exact relationship among the four states. If all three states A, B, and C are necessary before state D can be reached, then the relationship among the three states preceding state D is logical AND (Figure 3a). If any one of the states A, B, or C is sufficient to reach state D, then the relationship among the three states is logical OR (Figure 3b).

States are related not only by their causal connections but also by constraint relationships among the abstracted objects associated with them. The abstracted objects are represented as schemas consisting of attribute-value pairs. (The three parts of Figure 2 illustrate how the *State Model* and the *Abstracted Object Model* and the actual objects in the VMT system relate to one another.) Each object contains information that allows the system to determine the values for that object's attributes using objects whose attribute values are already known. Constraints among states can then be specified by states sharing the same object or via the relationships among the attributes of the objects attached to the states. For example, each HYP object (see Figure 2b and 2c) has an attribute LEVEL. The relationships among the LEVEL attributes of the HYP objects attached to the states LOW-HYP-CREATED and HIGHER-HYP-CREATED is expressed by the following sets of constraints. The value of attribute LEVEL of object HYP attached to state LOW-HYP-CREATED is obtained by calling the function GET-LOWER-LEVEL with the value of attribute LEVEL of object HYP attached to state HIGHER-HYP-CREATED. Conversely, the value of attribute LEVEL of object HYP attached to state HIGHER-HYP-CREATED is obtained by calling the function GET-HIGHER-LEVEL with the value of attribute LEVEL of object HYP attached to state LOW-HYP-CREATED.

The abstracted objects either point to existing objects in the VMT system or specify characteristics of objects that should exist in the system. The ability to represent not only objects that already exist in the problem-solving system but also objects whose existence is necessary in order for the system to achieve a particular state allows the model to serve as the basis for a high level simulation of the underlying problem-solving system. This simulation is accomplished by propagating attribute values among the interrelated abstracted objects based on the causal relationship among the states.

In addition to reasoning about system behavior in terms of sequences of states, we also need to reason qualitatively about how system object attribute values are computed from the attribute values of other objects and from system control parameters. This requires modeling some of the internal computations performed by the problem-solving system. In order to model the problem-solving system at this level, we use a model very similar to the one used for modeling the behavior of the system. In this case, the states represent values of attributes of the system objects, values of controls parameters, and values of important intermediate states of the internal computation. The transition arcs represent how the value of a state is computed from the values associated with the states that precede it. We are currently using a simple causal model in which the arcs are labelled as either having an increasing or decreasing



PART A: States related by AND PART B: States related by OR

Figure 3: Logical Relationships among States in the Model.

effect on the value of the state that represents the result of the computation [1]. Two states are connected by an *increasing arc* if an increase in the value of one state causes an increase in the value of the other state. In some cases not shown in this paper, we also need to reason using the exact formula representation of the computation.

The states in the model can thus represent different aspects of the underlying VMT system. One of the attributes in the state schema is the STATE-VALUE attribute. This attribute can represent one of several aspects of the problem-solving system. In some cases we are interested in whether a particular intermediate state has been reached; i.e., is there an object in the VMT system that matches the characteristics of the abstracted object associated with that state. In these cases the STATE-VALUE is *true* if the object does exist, and *false* otherwise. In other cases we need to reason about the value of some attribute of a particular object and relate it to the value of the corresponding attribute of another object. For example, we need to reason about the relatively low rating of a hypothesis with respect to another hypothesis. In these cases the STATE-VALUES represent the relationships among two or more objects in the VMT system. The values of the STATE-VALUES attributes are then *low*, *high*, or *equivalent*.

The model is organized into *clusters* of states (Figure 4 illustrates three such clusters). Each cluster

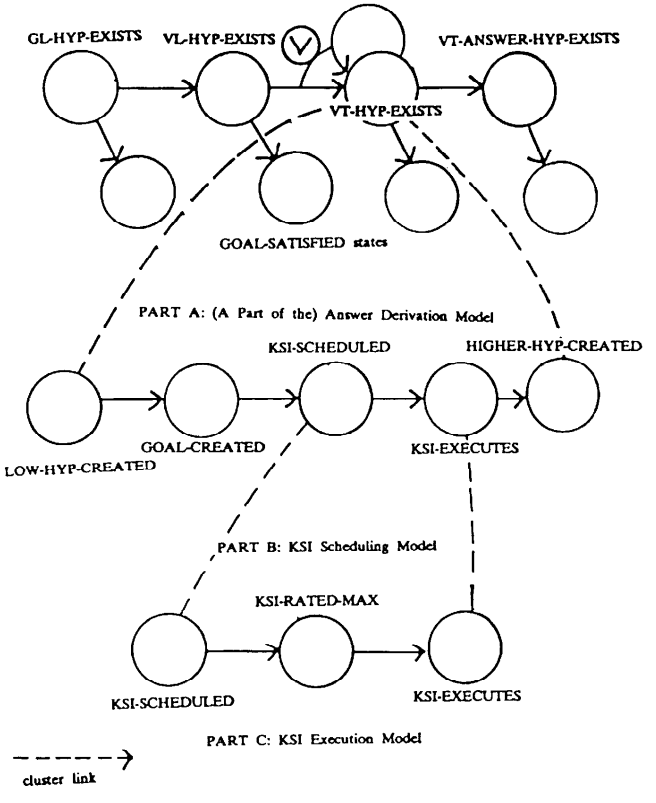


Figure 4: System Behavior Model Clusters.

represents an aspect of the system behavior at some level of detail. The representation is hierarchical in that only certain events are represented at any one level of the hierarchy. For example, the *Answer Derivation Model* represents only the answer hypotheses and their support structure in terms of intermediate hypotheses; vehicle track (VT) preceded by vehicle location (VL) preceded by group location (GL). It does not represent any of the knowledge sources scheduled and executed in the process. This information is represented in clusters at a lower level of the model hierarchy. Because of this hierarchical representation two states may be contiguous in one cluster while in fact a number of other states occur in between which are represented by a cluster at a lower level of the model hierarchy. Equivalent states in clusters at different levels of abstraction are connected via cluster links. Objects may be shared across the different clusters. This hierarchical structure allows fast focusing into the problem area during diagnosis by avoiding detailed analysis until the part of the model that is relevant has been identified.

The system model represents a subset of all the possible system behaviors, which we think is sufficient for detecting and diagnosing a significant number of faults.* We call this model the system behavior model (SBM). The SBM is used by both the Fault Detection module and the Fault Diagnosis module. The Detection module identifies a specific undesirable situation in the monitored system; i.e., a specific abstracted object along with an associated state. This state-object pair constitutes the symptom detected by the Detection module, which is passed on to the Diagnosis module. Diagnosis is accomplished by constructing a representation of the current system state, constructing a model of how this state was reached and comparing this with the correct system behavior as represented by the model. Any points of departure from this expected behavior are traced to the states at the lowest level in the SBM. These states are marked as primitive. A primitive state that is found to be false during diagnosis constitutes a reportable failure.

The current system state representation is constructed using information from the SBM and the VMT system data structures. The construction begins with locating the symptom state in the SBM. The predecessor states of this state are then found, along with their abstracted objects descriptions. First, the attributes of these abstracted objects are evaluated, using the constraint relationships between the existing abstracted object and the one being evaluated. Once these attributes have been evaluated, the Diagnosis module looks for the corresponding objects in the VMT system. If such objects are found, they are linked to the abstracted object. Finally, for each abstracted object the corresponding state is created and the STATE-VALUE

* The system model could be extended to represent the code level of the VMT system. However we have not found it necessary to represent the VMT system at such a low level of detail in order to effectively reason about its behavior.

attribute is evaluated. Depending on the type of state and its value, the type of reasoning may now change. The next paragraph describes the different types of reasoning.

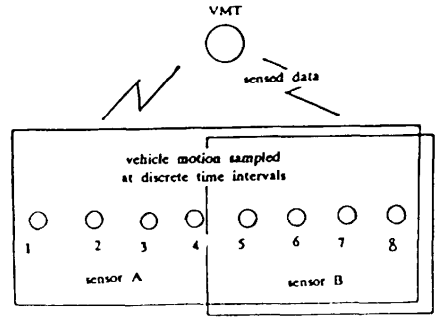
The underlying mechanism for all the different types of diagnostic reasoning is bi-directional constraint propagation, which begins at one or more state-object pairs in the SBM whose values have already been determined. This constraint propagation makes possible sophisticated diagnostic reasoning. In the next section we show how the system model supports four different types of reasoning necessary to diagnose inappropriate parameter settings:

1. *Backward causal tracing*: given a particular state and its value the system can go back through the model and explain, in terms of the model states, why that state was reached.
2. *Comparative reasoning*: the system can compare two different objects and explain why they were different, in terms of the model states.
3. *Unknown value derivation*: the system can determine a value of an unknown state in the system model by finding the value which is consistent with the known values of the surrounding model states.
4. *Resolving inconsistencies*: having found two inconsistent objects, the Diagnosis module can decide which one is correct by comparing both objects to a model of an ideal or expected object.

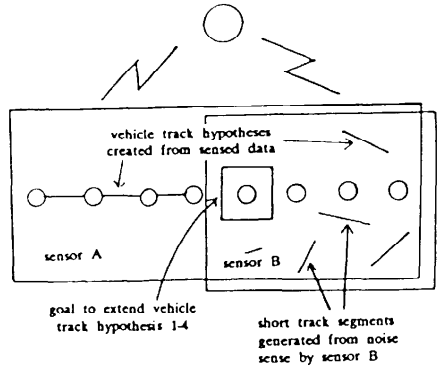
III AN EXAMPLE OF FAULT DIAGNOSIS

The following example (see Figure 5a) represents a scenario in the VMT system in which the system is receiving data from two input sources; sensors, A and B. The two sensors overlap, so some data are sensed by both, but because the system is more confident about sensor B the sensor weight parameters are set such that the data generated by that source are valued more than the data generated by sensor A. This results in the data from sensor B being rated high and the data produced by sensor A in the same area being rated low. In the example scenario the supposedly reliable source of data for the particular task (sensor B) does not in fact generate reliable data because it is malfunctioning. It is instead generating very short noise segments that cannot be incorporated into a single vehicle track. Because sensor B's sensor weight parameter has such a high value, these short noise segments are very highly rated. The goal of the diagnosis is to recognize that sensor B is malfunctioning and change the sensor weight parameters so that the systems begins to process data generated by sensor A.

A vehicle is moving through the monitored area, from left to right, generating signals at locations 1 through 8 (see Figure 5a). Sensor A senses all 8



PART A: Diagram of the signals generated by the moving vehicle (locations 1 through 8) and the sensor layout. The sensors send the sensed signals to the processing node.



PART B: After some time, the system generates a vehicle track (VT) hypothesis connecting locations 1 through 4 sensed by SENSOR A. It also generates several short track segments which are the result of the noise generated by the faulty SENSOR B.

Figure 5: Fault Scenario.

locations but, becuse of the sensor-weight parameter, locations 5 through 8 are rated low. Sensor B, because it is malfunctioning, is not sensing the vehicle signals but rather is generating very highly rated noise segments. The VMT system generates a vehicle track (VT) hypothesis connecting locations 1 through 4 based on the strong data from sensor A (see Figure 5b). As a result of sensor A's data being weighted low in the area where signals 5 through 8 appear, sensor B malfunctioning, and sensor B's sensor weight parameter being high, the knowledge source instantiation (KSI) that would extend the partial track to include the location in time 5 is rated low.* Because the short segments of noise generated by sensor B are rated high, they cause the scheduling of knowlege sources which are highly rated. The system queue has a number of these highly rated KSIs that delay the execution of the low rated KSIs which would extend the true vehicle track hypothesis. As a result, the system spends all its time forming short segments from the noise signals and the true vehicle track remains unextended.

* A KSI rating is a function of, among other things, the input data.

This situation can generate a number of symptoms. Due to lack of space we will illustrate the diagnosis by pursuing only one of the symptoms. The symptom we pursue here is a highly rated goal, VMT-GOAL#1, which represents the system's intent to extend the existing vehicle track hypothesis connecting locations 1 through 4 to include location 5 (see Figure 5b). This goal has remained unsatisfied for a long time and has therefore been selected by the Fault Detection module as a representative symptom. Diagnosis begins with the arrival of the symptom from the Detection module. A symptom consists of a state-object pair; the unachieved state is GOAL-SATISFIED and the abstracted object is GOAL-OBJECT, which points to the object VMT-GOAL#1 in the VMT system.

First, the SBM cluster that contains the state GOAL-SATISFIED and its associated abstracted objects must be located. This is the *Answer Derivation Model* cluster. The relevant objects and states in this cluster are evaluted, using the constraint expressions in the SBM and the already evaluated attributes of the symptom state and its object. The values of the states in this cluster can be either *true* or *false* depending on whether objects of the desired characteristics exist in the VMT system or not. In this case the state GOAL-SATISFIED is false because the associated object (VMT-GOAL#1) has not been satisfied in the VMT system (i.e., there is no vehicle track hypothesis connecting locations 1 through 5). We continue *backward causal tracing* through the SBM model to the state preceeding the GOAL-SATISFIED state: the state VT-HYP-EXISTS and its associated object, VT-HYP. The attribute values of this object are determined from the attribute values of the object VMT-GOAL#1 using the constraint relationships described in the previous section. The state VT-HYP-EXISTS evaluates to false, since no VT hypothesis of the desired characteristics exists in the VMT system. The reasoning continues backwards through the SBM attempting to find the first state that evaluates to true (i.e., the last point where desired system behavior stopped). Because a vehicle track can be formed from a shorter vehicle track or a set of vehicle locations (VL) the state VT-HYP-EXISTS is preceeded by the states VT-HYP-EXISTS* or VL-HYP-EXISTS. The objects associated with these states are VT-HYP and VL-HYP respectively. Again, we look for the associated objects in the VMT system in order to evaluate the states. In this case the objects are track fragments containing locations 1 through 5, or the locations 1 through 5 themselves, which could lead to the desired hypothesis. This brings us to another instantiation of the state VT-HYP-EXISTS and object VT-HYP, this time with the hypothesis connecting locations 1 through 4. Because such a hypothesis does exist in the VMT system, this state evaluates to true. This is where the generation of the vehicle track that would satisfy the goal VMT-GOAL#1 stopped. The evaluated model is in Figure 6a.

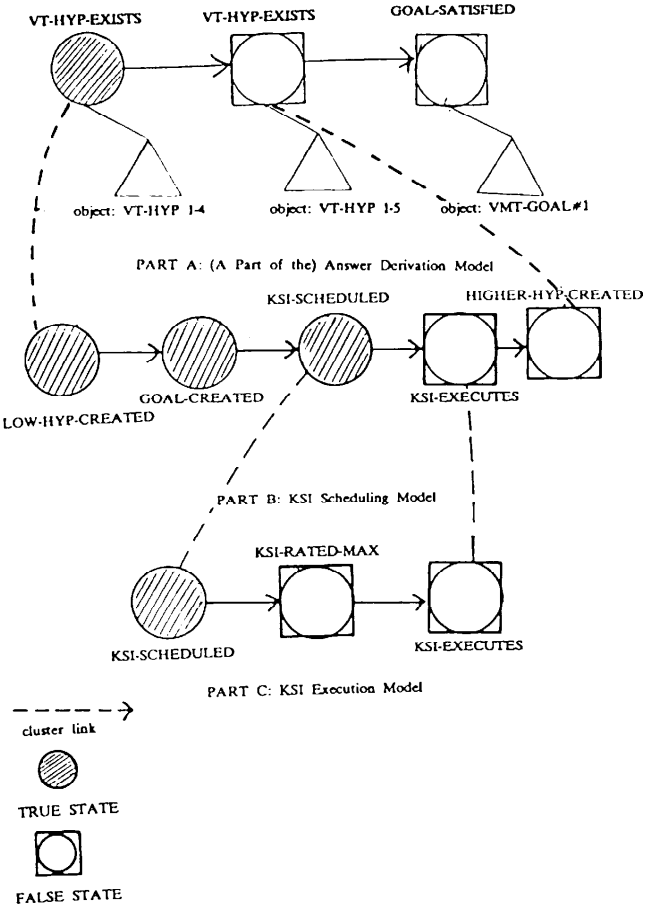


Figure 6: Evaluated System Model.

At this point we cannot continue reasoning using the *Answer Derivation Model* cluster because it does not represent the events occurring in between the last true state (VT-HYP-EXISTS; VT hypothesis connecting locations 1 through 4) and the first false state (VT-HYP-EXISTS; VT hypothesis extending the hypothesis 1-4 through location 5). Anytime such a true-state/false-state pair is found, we must find the cluster which represents the states occurring between those two states. The cluster pointed to by the VT-HYP-EXISTS state is the *KSI Scheduling Model* cluster shown in Figure 4b.

We continue determining the types of objects and evaluating the states. The result is the evaluated model in Figure 6b. We find another gap in the expected processing: the KSI that would produce the desired hypothesis was scheduled but did not execute. Again, following the cluster links, we switch to a cluster that describes in more detail what occurs in between the true state (KSI-SCHEDULED) and the false state (KSI-EXECUTES). This is the cluster *KSI Execution Model* in Figure 4c. We eventually arrive the state KSI-RATED-MAX. This state represents the fact that a KSI must be

* The state VT-HYP-EXISTS represents all track hypotheses up to some fixed track length. Therefore it is a reflexive state, pointing back to itself.

* Comparative reasoning contains many complexities which we cannot go into in this paper. For more detailed description of the types of reasoning mentioned in this paper see [10].

rated the highest of all the KSIs on the queue in order to execute. This state is false since the KSI that could extend the 1-4 VT hypothesis is rated low with respect to the other KSIs on the queue. The evaluated model is in Figure 6c.

The state KSI-RATED-MAX is a different type of state. Unlike the states mentioned so far, which represent the existence of some object in the VMT system, the state KSI-RATED-MAX represents a *relationship among a group of objects*; in this case, the relationship among the knowledge source instantiations on the scheduling queue. Whenever this type of a state is reached, the system switches to *comparative reasoning*.* This involves comparing some attributes of two objects in the system: one that achieved a desired state (in this case, the KSI that is maximally rated) and one that did not (in this case the low rated KSI that would extend the VT hypothesis 1-4 to include location 5). The system builds a model of how those objects were created and attempts to discover what differences along the object creation paths were responsible for the different outcomes. Two slots in the state schema are important here: the ACTUAL-VALUE slot, which represents the value of the attribute of interest, and the RELATIVE-VALUE slot, which represents the relationship among the ACTUAL-VALUES of the two objects in the parallel investigation. In this type of reasoning the states do not represent the existence or non-existence of some object but rather the relationship among the values of a particular attribute of some object (for example the rating of a knowledge source or a hypothesis) as compared to the corresponding attribute of the other object in the parallel investigation. In this case the relevant attribute is the RATING attribute of the KSI object. The two objects being investigated here are the two KSIs (the low rated KSI to create a hypothesis connecting locations 1 through 5 and the KSI which is rated the highest on the scheduling queue). We investigate, in parallel, how the ratings of the two KSIs were derived in an attempt to identify what caused the lower rating of the KSI that would extend the 1-4 track.

We first switch to a cluster where the attribute of interest (KSI-RATING) is represented by a state. This is the *KSI and Hypothesis Rating Model* in Figure 7. Because we are investigating two objects we must instantiate two copies of this cluster. One copy will represent the creation of the low rated KSI that would extend the VT hypothesis through location five (we will call this the *low ksi path*). The other will represent the creation of the highest rated KSI on the queue (we will call this the *high ksi path*). We begin with the state KSI-RATING. Because the rating of the KSI of interest is lower than the highest rated KSI we assign the value low to the RELATIVE-VALUE attribute of the state representing the relationship among the two values. We go back through the SBM and find that what determines a KSI rating is the DATA-COMPONENT-RATING of the KSI. We compare the data components of the two KSIs and again find that the DATA-COMPONENT-RATING of the low-rated KSI is lower than the corresponding DATA-COMPONENT-RATING of the high-rated KSI. We continue evaluating the model for the derivation of the KSI rating for both KSIs, via the KSI data components at various levels of abstraction (vehicle location, VL, preceeded by group location, GL, preceeded by signal location, SL) arriving finally at a point that represents how the sensor weights and the strength of the data signal determine the value of the sensed signal for each sensor.

Because the signal location rating on the *low ksi path* is lower than the signal location rating on the *high ksi path*, the value of the state SL-HYP-RATING for the low-rated KSI is low. We reason that in order for this value to be lower than the corresponding value in the *high ksi path*, the two objects that influence this value (sensed-value by sensor A and sensed-value by sensor B) must be rated lower than the corresponding objects on the other path. When we enumerate the relationships among the two pairs of sensed-values we get four relationships:

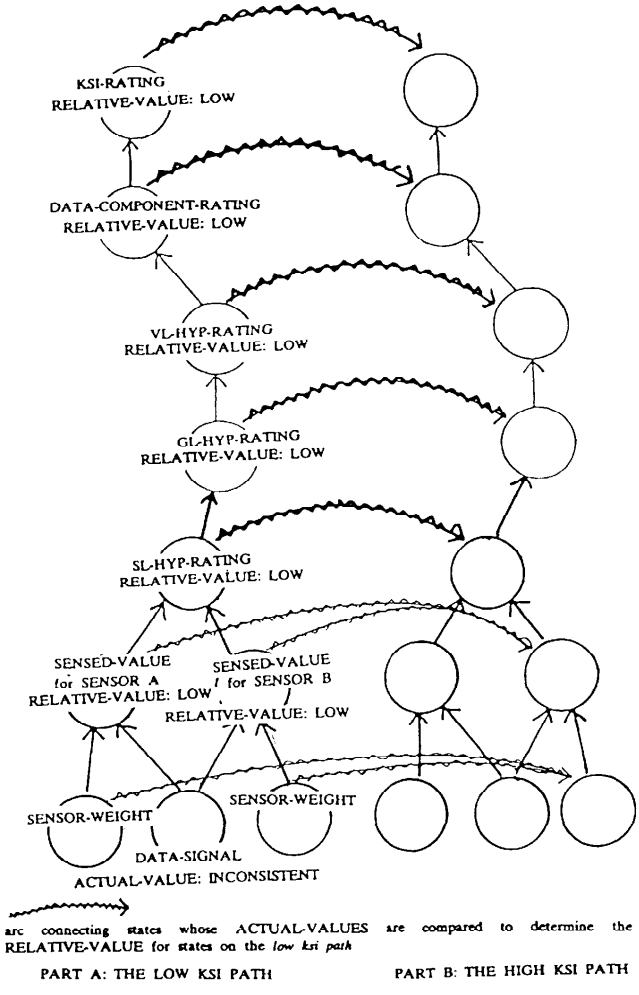


Figure 7: Parallel Investigation of two KSI Rating Derivation Paths.

1. Sensed-value for sensor A on the *low ksi path* < sensed-value for sensor B on the *high ksi path*.
2. Sensed-value for sensor A on the *low ksi path* > sensed-value for sensor A on the *high ksi path*.
3. Sensed-value for sensor B on the *low ksi path* = sensed-value for sensor A on the *high ksi path*. (They are both 0 because no signal was sensed at that place by the other sensor.)
4. Sensed-value for sensor B on the *low ksi path* < sensed-value for sensor B on the *high ksi path*.

In this case the RELATIVE-VALUE attribute of the state SENSED-VALUE for SENSOR A can have two values, depending on which of the corresponding sensed values in the other path we compare the state to: the values are *low* for case 1 above and *high* for case 2 above. Because we are trying to determine why the SL-HYP-RATING is lower, we follow paths to any states that contain a lower relationship. In this case, both the state SENSED-VALUE of SENSOR A and the SENSED-VALUE of SENSOR B contain a lower relationship so both are followed in parallel.

We have two paths to follow now: investigating why the SENSED-VALUE for SENSOR A was low with respect to SENSED-VALUE for SENSOR B in the *high ksi path* investigation and investigating why the SENSED-VALUE for SENSOR B was low, again with respect to SENSED-VALUE for SENSOR B in the *high ksi path* investigation. We first follow the path from state SENSED-VALUE for SENSOR A backwards. We reason that the sensor weight was low, the data signal was low, or both. We then find that value for SENSOR-WEIGHT for SENSOR A is indeed low compared to the SENSOR-WEIGHT for SENSOR B. Because this state is a primitive state (no transition or cluster arcs connect it to any other part of the model), we can report this finding as one fault responsible for the low KSI rating that led to the original symptom. We have found one problem that explains the low KSI rating but the investigation is not complete. We still need to find the value for the state DATA-SIGNAL and follow the path of low SENSED-VALUE by SENSOR B. This latter path also leads to the state DATA-SIGNAL since it is one of the predecessor states of the state SENSED-VALUE for SENSOR B.

Since there is no way of knowing what the actual data signal was, we must employ the *unknown value derivation* type of reasoning where an unknown value is determined by examining the values of the neighboring states. This type of reasoning is necessary anytime the state value cannot be determined from the problem-solving system's data base. In this type of reasoning the ACTUAL-VALUE (or STATE-VALUE) attributes of the states represent the value that is derived by looking at the surrounding states. Depending on the types of values represented by those states, this value can be either the value the states agree on or inconsistent if contradictory values can be determined from the surrounding states. The unknown state is DATA-SIGNAL. We attempt to derive the value for this state, which represents the actual value of the data signal in the environment, by examining the ACTUAL-VALUE slots in its surrounding states: SENSOR-WEIGHT for both sensors and SENSED-VALUE for both sensors. In fact we cannot find a consistent assignment for all these states. According to sensor A the value sensed is low; according to sensor B, no value is sensed at all. The value for the state DATA-SIGNAL is therefore INCONSISTENT. In a case where an inconsistency is discovered among two objects in the VMT system we have to use *inconsistency resolving reasoning* in which we compare the two objects (in this case the two disagreeing sensors) with a model of the expected behavior of that object (a sensor) and try to determine which one is correct. In this case we compare the characteristics of each of the two sensors with the characteristics of an ideal sensor which produces correlated data. We determine that data from sensor A is well correlated (all data fits into one track) whereas data from sensor B is only correlated for at most 2 location track segments. We therefore conclude that sensor B is faulty.

We have now found both reasons for the initial symptom (unsatisfied goal): the faulty sensor B in conjunction with the low SENSOR-WEIGHT parameter for sensor A.

IV STATUS AND FUTURE RESEARCH

The basic model and the constraint propagation mechanisms have been implemented. We are currently extending the system to handle the comparative reasoning. Currently the system behavior model represents only the system behavior. It does not make an attempt to represent the reasons for the expected behavior in terms of the system architecture (e.g., a goal represents the intent to produce a hypothesis in the goal's area) or in terms of the assumptions about the domain (e.g., the characteristics of goals based on hypotheses that led to them). We believe that such deeper models of both the architecture and the domain would increase the FDD system's expertise by allowing it to detect more subtle errors (e.g., redundant satisfaction of goals) and to detect a wide range of faulty assumptons about the task domain. An example of the latter case is having a model of how the goal characteristics depend on the hypothesis characteristics, for example, the maximum acceleration of a vehicle and its turning radius. We also believe that such a deeper model of the problem-solving system could serve as a knowledge-base that the system could use to automatically generate the complex criteria necessary for fault detection and the knowledge needed to implement the Strategy Replanning module.

We feel that meta-level control based on the Fault Detection/Diagnosis paradigm represents a new approach to introducing more sophisticated control into a problem-solving system. In addition, the system can be of great help in debugging complex problem-solving systems. It also presents interesting issues in modeling and reasoning about a problem-solving system.

ACKNOWLEDGEMENTS

We would like to thank Daniel Corkill for his help in developing the ideas and implementation presented in this paper.

V REFERENCES

1. Stephen E. Cross. Qualitative sensitivity analysis: A new approach to expert system plan justification. Technical Report, AI Lab., Dept. of Electrical Engineering, Air Force Institute of Technology, Wright-Patterson AFB, 1983.

2. Daniel D. Corkill, Victor R. Lesser, and Eva Hudlicka. Unifying data-directed and goal-directed control: An example and experiments. In **Proceedings of the National Conference on Artificial Intelligence**, pages 143-147, August 1982.

3. R. Davis. Meta-Rules: Reasoning about control. *Artificial Intelligence*, 15:179-222, 1980.

4. R. Davis, H. Shrobe, W. Hanscher, K. Wieckert, M. Shirley, and S. Polit. Diagnosis based on descriptions of structure and function. In *Proceedings of Proceedings of the National Conference on Artificial Intelligence*, pages 137-142, August 1982.

5. M. Genesereth. Diagnosis using hierarchical design models. In *Proceedings of Proceedings of the National Conference on Artificial Intelligence*, pages 278-283, August 1982.

6. Michael R. Genesereth and D.E.Smith. Meta-Level Architecture. Heuristic Programming Project Memo HPP-81-6, Stanford University, December 1982.

7. Barbara Hayes-Roth and Frederick Hayes-Roth. A cognitive model of planning. *Cognitive Science*, 3(4):275-310, October-December 1981.

8. Frederick Hayes-Roth and Victor R. Lesser. Focus of attention in the Hearsay-II speech understanding system. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 27-35, August 1977.

9. Eva Hudlicka and Victor Lesser. Design of a knowledge-based fault detection and diagnosis system. In *Proceedings of the 17th Hawaii International Conference on System Sciences*, Vol 1., pages 224-230, January 1984.

10. Eva Hudlicka and Victor Lesser. Diagnostic reasoning in fault detection and diagnosis for problem-solving systems. COINS Technical Report (in preparation).

11. Victor Lesser and Daniel D.Corkill. The Distributed Vehicle Monitoring Testbed: A tool for investigating distributed problem solving networks. *AI Magazine* 4(3):15-33, Fall 1983.

12. B. Smith. Reflection and Semantics in a Procedural Language. Artificial Intelligence Laboratory Memo AI-TR-272, MIT, January 1982.

13. Robert Wilensky. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, 5(3):197-233, July-September 1981.