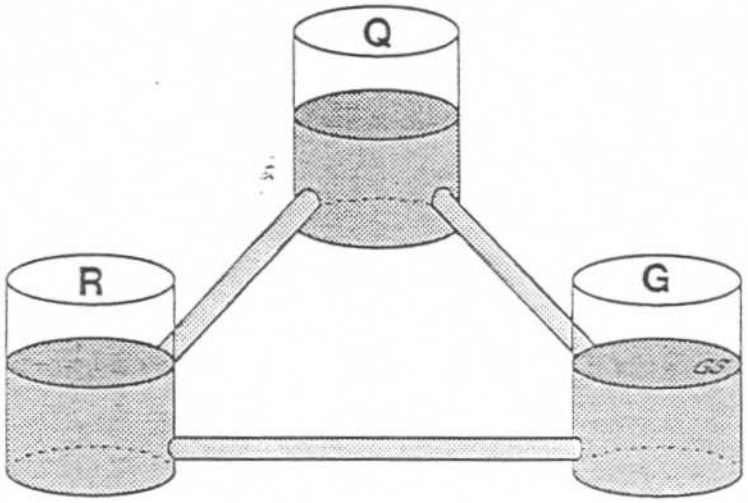
DEPARTMENT OF COMPUTER SCIENCE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN



QUALITATIVE REASONING GROUP

REPORT NO. UIUCDCS-R-91-1699 UILU-ENG-91-1745

DIAGNOSIS AS FAILURE UNDERSTANDING

by

John W. Collins

May 1991

Diagnosis as Failure Understanding
(preliminary report)

John W. Collins

Qualitative Reasoning Group
Beckman Institute
University of Illinois at Urbana-Champaign

April 11, 1991

Abstract

This paper reports on research in progress, investigating the use of process-centered qualitative models in the diagnosis of continuous variable systems. I view diagnosis as a problem of minimally perturbing a faulty model of a system until it matches observed behavior. I claim that qualitative models can account for observed symptoms without relying on explicit fault models, thereby increasing robustness for novel faults. Both the mechanism of failure and the resulting state of the failed system can be modeled in terms of their underlying active processes. By focusing on the process structure rather than the physical structure, the search space is both smaller and more focused, since only those processes which could influence the system in the observed directions need to be considered. Several diagnostic strategies are being explored within this framework. An implementation is in progress, building on a modified ATMS and an incremental qualitative envisioner. The kinds of systems I am examining are taken from the domain of thermodynamics, and include piston and gas-turbine engines, a steam boiler and a thermal control system.

1 Introduction

"Diagnosis begins with a failure to understand."

We live in a manufactured world. Every day of our lives, we are surrounded by machines intended to increase our productivity, our mobility, and the quality of our lives. In addition to death and taxes, there is one undeniable truth in this synthetic world: machines break down. Whether it is due to planned obsolescence, misuse, poor design or unavoidable wear or fatigue, machines require almost continual repair and maintenance.

The obvious first step in repairing a broken device is to understand what is wrong with it. Unfortunately, as machines become more complex, there are more ways for them to fail, and the failures become more difficult to understand. The general act of understanding the nature, location and extent of failure in a misbehaving machine is called *diagnosis*.

This research investigates the application of process-centered qualitative models to understanding novel failure modes—both in terms of the mechanism of failure and the resulting state of the failed system. By viewing the problem as residing in the model of the system rather than the system itself, and recognizing that the system continues to obey the laws of nature, it is possible to apply the same knowledge to derive expectations and to determine why they fail to be met.

1.1 An Example

Consider the following scenario. You are driving your car down I-57, when suddenly your engine sputters and dies. You pull over and coast to a stop, thinking the worst. Almost instinctively, you look at your fuel gauge and see that it shows half a tank. As you collect your wits, you begin to consider your next move.

Without knowing what you're looking for, you get out of your car and open the hood; nothing stands out as obviously wrong. You ask yourself what an engine needs to run: "spark, fuel, timing,..." You decide to pull a spark plug wire to check for a spark while your spouse turns the key—the shock you receive convinces you that the problem is elsewhere. You remove the air filter to examine the carburetor; you pump the throttle and watch for gasoline from the accelerator jets: you see no gas.

A lack of fuel at the carburetor would certainly explain why your engine died. But how can that be? Maybe you really *are* out of gas; the gauge might be stuck. You turn on the key and watch the gauge slowly rise to 1/2. The float in the tank could be stuck; you can't think of an easy way to test that. But you know your car's gas mileage, and there *should* be a half tank left—unless there's a fuel leak, or your fuel economy has decreased significantly. A fuel leak and a stuck float is one possible explanation, though not a very likely one. Another possibility is that there *is* fuel at the carburetor, and the accelerator pump has failed; but that alone would not cause the engine to

stall. You dismiss these possibilities, because they involve too many coincidences. What else could it be?

You decide to explore the fuel system in more detail. You remove the fuel line from the inlet of the fuel pump and observe gasoline pouring out. Now you know there's fuel in the tank but none at the carburetor. After replacing the inlet line you remove the outlet line from the fuel pump and crank the engine: no gasoline is seen. The fuel pump *must* be the culprit! After a few hours and a new fuel pump, you're on your way.

This example was drawn from a personal experience; I still remember the thought processes and actions which eventually led to the correct diagnosis. I include it here because it illustrates several important aspects of the diagnostic task.

Diagnosis involves search. Troubleshooting a misbehaving machine requires exploring a vast space of possibilities, looking for possible explanations for the unexpected behavior. Knowledge about the system and the domain help guide this search down promising paths. This may be very specific, shallow knowledge such as a memory of a similar problem with this particular artifact; or it may be very general, deep knowledge of the domain, such as the concepts of conservation of matter and energy.

Some faults are more likely than others. There may be multiple diagnoses which are consistent with what is known about the failed artifact. Choosing one path among many must be based on estimates of relative likelihood. A search path may be abandoned temporarily, even though it is still viable, to look for a more reasonable explanation. A candidate diagnosis involving coincidental multiple faults will generally not be considered until simpler explanations have been ruled out.

Diagnosis involves multiple levels of reasoning. Efficient diagnosis requires reasoning about the failed system on multiple levels of detail. In the example above, the fuel system was initially treated as a black box, whose internals were irrelevant to the initial reasoning steps. Later, when conflicting observations arose concerning the fuel system, it was examined in greater detail as a series of individual components. Knowing how to represent a system at multiple levels of abstraction and when to explore a subsystem in greater detail are central issues in diagnostic research.

Diagnosis involves action as well as inference. Diagnosis is not done from an armchair; it requires interacting with the failed system—observing, probing, even modifying it in order to understand its condition. Performing measurements is only one of many ways to obtain information about a failure. Of all the actions in the example above, checking the fuel guage was the only action resembling a measurement. Deciding what to look for and what to modify are two important aspects of this research.

1.2 Background

Diagnosis is a knowledge-intensive cognitive task, and has been an area of active research in AI for the past two decades. Early attempts at automating diagnosis were based on shallow knowledge,

such as associations between symptoms and failure modes. Expert systems such as MYCIN [23] showed impressive diagnostic abilities within their intended domain, but were quite brittle when applied to novel systems or modes of failure.

Recognizing these limitations, some researchers [4,14] began to investigate the use of deep knowledge in diagnosis. In certain well-understood domains such as electronics, models of individual components and their failure modes were developed and used to automatically compose system-level models from schematic-level descriptions. Qualitative models were shown to be sufficiently rich for much of this style of reasoning. Model-based diagnostic systems were able to reason effectively with novel systems constructed from the known component types. However, they still suffered from the need for explicit fault models. That is, they were brittle when it came to novel failure modes.

A technique called *constraint suspension* [5] was introduced to overcome the dependency on explicit fault models, while still identifying components which *might* be misbehaving. If suspending the constraints associated with a suspected component eliminates the conflict between observations and expectations, then that component may be responsible for the observed symptoms. This approach extends to novel modes of failure which were not anticipated by the model builder. GDE [7] reimpliments constraint suspension using an assumption-based truth maintenance system (ATMS), and can handle multiple faults efficiently. Constraint suspension works well in domains such as digital electronics, where signals propagate only in one direction and interactions are minimal. It is of little help in systems involving feedback, for example, since the failure of any one component could potentially account for any anomalous behavior. Another limitation is that symptoms are not actually *explained*. Without explicit fault models, there is no way to verify that some physically-realizable mode of failure in the suspected components could actually account for the observed behavior.

1.3 Overview

I have identified two major weaknesses common to existing approaches to automated diagnosis:

1. *The inability to account for symptoms arising from novel failure modes;* and
2. *The lack of a general strategy for active investigation of the failed artifact.*

My research aims at addressing these two issues.

Process-Centered Diagnosis My approach accounts for unanticipated modes of failure by applying deep-level qualitative models in two novel ways:

Understanding the failed artifact: The state of some artifact after it has become inoperative may be understood in terms of its underlying process structure. For example, a leak in a fluid system may be seen as an instance of a fluid flow process. No special fault model for leaks is required, so long as fluid flows are already part of the domain knowledge. In general, a

fault may activate new processes or deactivate existing ones. By initially focusing on the active process structure rather than the underlying physical structure of a failed artifact, the search space is greatly reduced. The search for new process instances is further focused by knowledge of which quantities are changing unexpectedly, given that each process type only affects certain types of quantities.

Understanding the failure mechanism: Process-centered models may also be applied to understanding the *mechanism* of failure—that is, how and why a failure occurs. For example, a leak may be caused by corrosion or fracture, among other possibilities. Identifying the possible mechanisms of failure is important for verifiction and evaluation of a fault hypothesis, particularly when multiple dependent faults are involved.

Active Diagnosis As demonstrated by the above example, diagnosis often involves actively acquiring additional information about a failed system. With the exception of measurement planning, this area has been neglected by other researchers. My research explores what I call *active diagnosis*: the selection of actions to be performed to a failed artifact for the purpose of obtaining useful information about the nature of the failure. These actions can take a variety of forms, some of which have been categorized according to their expected consequences.

The remainder of this paper is organized as follows. Section 2 reviews research in qualitative reasoning and shows how it relates to this research. Section 3 provides a detailed account of the diagnostic task and how I plan to approach it. Section 4 describes implemented components for supporting the diagnostic reasoning, as well as plans for further implementation. Section 5 illustrates my approach to diagnosis on an example drawn from the aeronautics domain. Finally, Section 6 summarizes the goals and overall approach of the research.

2 Qualitative Reasoning

Qualitative reasoning attempts to describe or predict the continuous behavior of physical systems, without manipulating actual numbers. Typically this is accomplished by partitioning the range of values of a quantity into a small number of interesting subregions. A common example of such a partitioning is the sign operator, which maps a quantity into three regions: *positive, negative* and *zero*. A generalization of this approach allows comparisons between any two quantities (instead of just comparisons with zero). The leverage of qualitative reasoning comes from making only those distinctions which are relevant to the problem at hand.

Unlike numerical simulation, qualitative reasoning provides the ability to reason from incomplete information, and with potentially fewer computations. In addition, qualitative reasoning provides a psychologically plausible account for much of human reasoning. People clearly do not compute exact solutions or run numerical simulations in predicting or understanding an outcome. Instead, we

often describe behaviors in qualitative terms, without reference to magnitudes or units. Yet we are able to predict behaviors and postulate the underlying mechanisms responsible for an observation. Qualitative reasoning plays a major role in human reasoning, and thus holds promise for knowledge intensive problems like diagnosis.

My research investigates the use of qualitative models in diagnosing continuous-variable systems. More specifically, I am assessing the potential leverage afforded by process-centered qualitative models in understanding both the mechanism and resulting state of a novel failure. The following section reviews Forbus' Qualitative Process theory, which is the basis for the qualitative models used in my research.

2.1 Review of Qualitative Process Theory

Qualitative Process (QP) theory [11] is based on the tenet that all change results from the action of *processes*. The possible types of processes are defined in terms of the conditions under which they are active and the conclusions which follow.

A second tenet of QP theory is that qualitatively interesting changes in behavior coincide with changes in certain inequalities. For example, the flow between two containers stops exactly when their levels become equal. Thus the qualitative state of a system may be described in terms of a collection of inequality relations. By considering how these relations can change over time, it is possible to simulate the sequence of qualitative behaviors of the system being modeled.

QP theory defines a language for expressing a general theory for some domain, as well as a procedure for automatically deriving a causal model for a given scenario in that domain. Such a domain theory offers *composability*, in that an unbounded number of distinct scenario configurations can be modeled in terms of their underlying process types.

Quantities, Numbers and Inequalities Quantities play a central role in QP theory. A *quantity* represents some continuous parameter which is relevant to the model. A quantity belongs to one or more individuals, and only exists when all its owners exist. Associated with each quantity are two *numbers*: its amount and derivative (w.r.t. time). Numbers in turn have a sign and a magnitude. An *inequality* relates two numbers by defining the ordering between them. Possible values for an inequality relation are: $\{>, =, <, \oslash\}$; where $\oslash$ represents *unrelated*, indicating that one of the numbers does not exist.

Objects, Attributes and Relations Objects are defined in QP theory as belonging to one or more *entity classes*. Each entity class has a set of attributes which are inherited by its instances. Typical attributes include the quantities of an object, the causal and inequality relations between them, and more general entity classes to which the object belongs. Connectivity relations between two or more objects define the structural connections of the scenario.

Views and Processes Views and processes are conditional descriptions of state and behavior, respectively. An instance of a view or process exists for each combination of objects matching the *individuals* of the corresponding *view type* or *process type*, as defined by the domain theory. An existing view or process instance becomes *active* when its *preconditions* and *quantity conditions* are satisfied. Preconditions represent modeling assumptions or controls which are manipulated by some external agent (such as a manually-operated valve). Quantity conditions are a set of inequality relations required by the view or process. As a consequence of the active instance, the *relations* of the view or process become true.

Causal Influences Processes differ from views in that processes have *influences*. An active process instance directly influences one or more quantities; the degree of influence is equal to the *rate* of the process. For example, a liquid flow process directly influences the amounts of liquid at the source and destination of the flow. It is possible for more than one process to directly influence the same quantity; the derivative of the influenced quantity is equal to the sum of the rates of the influencing processes. Processes and their influences are viewed as the causal origin of all change; if there are no active processes, then all quantities must be constant.

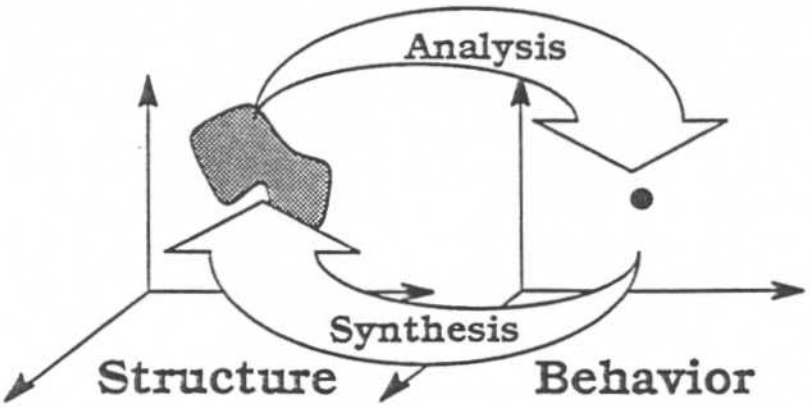
Quantities can be indirectly influenced by a process, through chains of *qualitative proportionalities*. As with direct influences, qualitative proportionalities may be either positive or negative. The relation: $Q_1 \propto_{Q+} Q_2$ states that Q_1 will change in the same direction as Q_2 when all other determiners of Q_1 are constant. In other words, Q_1 is monotonically increasing in Q_2. Likewise, $Q_1 \propto_{Q-} Q_2$ has Q_1 monotonically decreasing in Q_2. In both cases it is implicitly assumed that a change in Q_2 *causes* a change in Q_1. Therefore the qualitative proportionalities must run outward from directly influenced to indirectly influenced quantities. Since these relations are viewed as imputing causality, loops among qualitative proportionalities are not allowed.

Qualitative Modeling in Thermodynamics The modeling language defined by QP theory has allowed the accumulation of a significant domain theory for thermodynamics [2]. In my research I plan to apply this domain theory to diagnosing thermodynamic systems. This will require extending the theory to allow hierarchical representation of complex structures. The domain theory will also be augmented with various fault models; while my approach to diagnosis does not rely on them, explicit fault models make diagnosis much easier, and should be used when available.

3 A Theory of Understanding Failures

Diagnosis may be viewed either as a task to be accomplished, or as a procedure for accomplishing that task. Designing a diagnostic procedure requires a careful analysis of the diagnostic task. These two aspects—the "what" and "how" of diagnosis—are presented in turn.

Figure 1: Diagnosis is a Synthesis Problem



3.1 Understanding the Diagnostic Task

Diagnosis is the act of identifying a set of faults given a set of symptoms. More specifically, diagnosis is the act of inferring the structural deviations of a failed system which explain an observed deviation from expected behavior. Diagnosis is generally performed as a subtask of some larger goal—typically repair or damage control. In either case, it is not always necessary or even desirable to narrow the set of candidate hypotheses to a single possibility. It may be that two or more candidates suggest the same corrective action, so that distinguishing between them is unnecessary. For example, if a problem in an electronic circuit has been isolated down to a single module which is cheaper to replace than repair, then it does not matter which component of the module has failed. In other cases different corrective actions may be taken simultaneously without interference or excessive cost, as a way of "covering all bets". These possibilities suggest that knowledge of the overall goals of the problem solver should be made available to the diagnostic reasoning component.

3.1.1 Diagnosis as Synthesis

In the most general view, diagnosis is an instance of a synthesis problem. Whereas analysis involves a mapping from structure to behavior, synthesis involves the inverse mapping from behavior to underlying structure (see Figure 1). Other examples of synthesis include design, planning, reverse engineering, and general understanding of observations.

Synthesis is generally harder than analysis, for several reasons. The mapping from behavior to structure is not always unique. There may be several different physical devices whose outward

behaviors are indistinguishable. The problem is aggravated by the lack of a complete behavioral description. In addition, the mapping from behavior to structure is generally not well understood. Synthesis usually requires a verification phase, where a candidate structure is analyzed to verify that it produces the desired behavior.

Diagnosis differs from other synthesis problems in that it attempts to reconcile a discrepancy between observations and expectations. More specifically, diagnosis involves a failure of some artifact to achieve its intended function. Diagnosis generally concerns a system which has worked at one time and has stopped working, or whose performance has degraded over time.[1] If the artifact worked in the past, then there is some basis for expecting it to continue to perform its intended function. Given that the laws of nature and the functional requirements of the artifact have remained fixed, it follows that the physical structure of the artifact has changed in some way.

3.1.2 Failure: Artifact vs. Model

When the behavior of a designed artifact deviates from that predicted by our model of it, the problem is traditionally viewed as residing in the artifact. I take the opposite view that the problem is in the model from which our expectations are derived. Specifically, the model has failed to keep up with changes in the artifact. Regardless of the severity of the "damage" to the artifact, it continues to operate "correctly", viewed from the perspective of the laws of nature. It is our expectations which need adjusting.

By recognizing that a failed or damaged system obeys the same laws as a fully functional one, we can apply the same knowledge to understanding failure as we apply to understanding behavior in general. Specifically, the knowledge used to produce our original expectations can be used again to explain why they were not met.

3.1.3 Failure: State vs. Mechanism

I view diagnosis as providing answers to two questions: "What *is* wrong?", and "What *went* wrong?". Each of these questions provides some leverage in identifying the nature of the problem. The first question involves the resulting state once the failure has occurred. Answering this question requires finding a model of the post-failure state of the artifact, capable of explaining its post-failure symptoms.

The second question involves the nature of the failure itself. It asks for a sequence of behaviors consistent with observations and leading from the original working condition to the post-failure condition. This question is secondary to the first question; it is asked as validation or comparison

[1] The term is occasionally applied to the debugging of a newly-designed system which fails to perform "up to spec"; this differs from the standard usage in that the problem may be with the design itself rather than with the artifact; that is, the expectations may be unfounded.

of candidate answers to the first question—or to prevent a repetition of the same failure after repair or in other systems of similar design.

As an example of a mechanism of failure, the development of a leak may be explained by any of a number of processes, including corrosion, expansion or shrinkage (of mating parts), or fracture. These processes provide possible answers to the second question. The answer to the first question in the case of a leak is that there is a fluid path and an active fluid flow process which were not previously known to exist.

Finding a mechanism capable of explaining why the failure occurred makes a candidate diagnosis much more plausible. In some cases it may be possible to determine that the failure *must* have occurred, given what we know about the environment which immediately preceded the failure; but this is the exception. Still, inability to find a plausible mechanism in a well-understood domain may cast considerable doubt on a candidate model of the failed state.

Some failures occur suddenly—one blink and it's over. In other cases failures occur so gradually that they go unnoticed. If the failure was missed the first time around, it may be possible to re-create the failure in a duplicate system in order to fully understand what went wrong. But imagine trying to duplicate the failure at Chernobyl; or the ten years of corrosion in a ship's boiler leading to its exploding. In any case, we generally do not have the luxury of replaying a failure again and again while trying different measurements or corrective actions. This makes finding the underlying mechanism of failure a difficult task—the space of possible failure mechanisms which could be occurring is too large to search without some guidance.

However, it is possible to carefully observe the failed system, while interacting with the system to test hypotheses. It is easier to determine the mechanism of failure after first understanding the resulting state of the failure. That is, the two questions above are most naturally answered in the given order. Knowledge of the resulting state helps to focus the search for a possible mechanism, more than the reverse.

3.2 Automating the Diagnostic Task

I view diagnosis as a problem in perturbing a faulty model of a system until it matches the observed behavior of that system. This follows the AI paradigm of *generate and test*, whereby candidates are suggested and then checked against a set of requirements. Efficient search requires that the generator make intelligent choices about which paths to consider first. Inconsistent paths should be rejected as early as possible, and the remaining paths should be ordered based on estimates of their probability of success. For instance, in searching for a model of the post-failure state of some artifact, it is reasonable to focus the search in the immediate vicinity of the original model. This assumes that the system has changed only slightly from its original working condition. Knowledge about possible mechanisms of failure can provide additional guidance in the search for a model of the post-failure state.

3.2.1 Qualitative Symptoms

Diagnosis begins with a behavioral description which does not match expectations. These deviations from expected behavior are called *symptoms*. I assume that symptoms are represented qualitatively, and are of one of the following forms:

Inequality Deviation: Some inequality relation was observed to be different than expected; this includes deviations in directions of change.

Magnitude Deviation: Some quantity or rate is observed to be larger or smaller than expected; such deviations may not be detectable from a purely qualitative model, but may still provide useful qualitative information.

Discrete Deviation: Some object or event is unexpectedly absent (or present). In some cases these may be represented as inequality deviations, where the mass of some object or the rate of some process has become zero (or non-zero).

3.2.2 Generating Failure Hypotheses

The simplest approach to generating candidate diagnoses would be to randomly pick some perturbation to the model, and then analyze the resulting model to see if it agrees with the observations. However, the presence of a qualitative model allows us to do better than this. Judicious use of the model allows the search to remain focused on reasonable possibilities.

The Role of Processes Processes provide an intermediate representation between structure and behavior. In order for a candidate model to match the observations, its process structure must be consistent with those observations. If some quantity is unexpectedly changing, then there must be some active process influencing it. Similarly, if a quantity is expected to change but does not, then either some process that was believed to be active is not active, or some other process is interfering with the known process.

The space of possible processes is smaller and more abstract than the space of all possible structures, so it is more efficient to find a valid process structure before speculating on the underlying structural details. In addition, processes can be indexed by their influences, so that only those processes capable of influencing some unexpectedly changing quantity need to be considered.

There are two ways to perturb the process structure of the model: removing an existing process, or adding a new process. These require different styles of reasoning, and are treated separately.

Removing an Existing Process A process which was active before failure may be suspected of inactivity, if removal of the process could explain an unexpected observation. The simplest case involves a quantity which was expected to change but is observed to be constant. It is reasonable to suspect that the responsible process has failed to be active for some reason; however it is possible

that the process is being countered by some competing process. For example, if a liquid flow was occurring between two containers prior to failure, and after failure both liquid levels are observed to be constant, it is likely that the flow has been obstructed in some way. Another case for suspecting a process of inactivity is when it competes with other processes to influence some wayward quantity. Removal of a process in competition may explain why a quantity is changing at a different rate or in the opposite direction than expected.

Adding a New Process It is generally more difficult to consider model perturbations involving the addition of previously unknown process instances, since there can be an arbitrary number of them. Still, there are only a fixed number of process *types*, and only those with the appropriate kinds of influences need to be considered.

The obvious case for speculating a new process is where a previously-uninfluenced quantity is unexpectedly changing; since all change is caused by processes, there must be some unknown process at work. For example, if the water level in some container is believed to be uninfluenced yet is observed to be dropping, then there must be some process (eg., a leak) which is negatively influencing the amount of water. On the other hand, if the pre-failure model contains competing processes, then removing one of them could also explain the change, as stated above. If the quantity was expected to change but at a different rate or in the opposite direction, then the addition of an active process provides a possible explanation.

Modifying the Scenario Description After a candidate process structure has been generated, the next step is to transform it to the underlying structural description of the scenario. We want to perturb the original description as little as possible while satisfying the requirements for the new process structure.

Deletions from the process structure generally can be achieved by removing a component from the scenario description, so long as it does not still participate in other ways. For example, a fluid flow process may be eliminated by removing the flow path. This does not imply that the physical pipe has vanished—rather, that the pipe is no longer behaving as a path (perhaps it is clogged).

Additions to the process structure require positing components to participate in the new process. These may be new objects which were not previously known to exist; or they may already be part of the scenario description, but filling unsuspected roles. For example, in the case of a leaking container, the container itself fills the role of the path for the fluid flow process. We knew about the container—we just didn't know it was also a path. Deciding whether to posit new objects or to use existing ones to fill needed roles is an open issue which this research will address.

3.2.3 Active Diagnosis

The underlying goal of diagnosis is to identify the nature of the fault, often as a first step in its repair. The search for a valid model of a failed system in the ideal case leads to a single consistent hypothesis. However in many cases there may be too many consistent fault hypotheses to consider them all, even though only one of them is valid. A general strategy for reducing the set of candidate hypotheses is to obtain additional information on the failed system. This involves interacting with the system in various ways, ranging from passive observation to disassembly and attempts at repair. I call this general process of interaction *active diagnosis.*

Performing Measurements Candidate hypotheses may be verified or refuted through additional observations or measurements of the faulty behavior. Some measurements are more easily obtainable than others, and some will provide more information than others. Ideally one should choose to perform the measurement with the highest expected informational gain for the least cost. Determining which quantities are measurable and at what cost requires detailed information about the failed system as well as the available measurement instruments and procedures. In lieu of this, I assume that each quantity has an associated measurability parameter indicating the cost of measuring its value.

Exposing Unobservables Performing certain measurements requires first modifying the system so as to make the desired quantity accessible. In some cases, partial disassembly may be required. In the fuel pump example from Section 1.1, the fuel line was disconnected from the fuel pump in order to reveal the presence (or absence) of gasoline inside. In so doing, the system was altered, making possible a flow of gasoline which was not occurring previously. In fact it was the consequences of this flow that were actually observed. It is not uncommon for such actions to yield observable consequences which provide useful information concerning the condition of the system.

Partial Repair One way to verify a failure hypothesis is to choose a modification which—given that the hypothesis is correct—will reduce or eliminate the deviation from nominal behavior. Typically this involves replacing or repairing a suspected faulty component, or providing a redundant system to operate in parallel with the suspected component. Behavioral improvement resulting from such a modification provides supportive evidence for those hypotheses which predicted the improvement, while eliminating those which did not. Similarly, if no improvement results from the change, then those hypotheses predicting improvement are ruled out.

In the case of multiple faults, this strategy may be repeated until no faults remain. This assumes that eliminating one of several faults will lead to an observable improvement in the system's behavior. This will not always be the case; in fact it is possible that the new or repaired components may become faulty due to interactions with other faults. For example, if a short circuit causes a

fuse to blow, then replacing the fuse will only yield a second blown fuse. However, if the failure is detectable and the replacement is not too costly, then the resulting gain in information may justify the cost.

Purposeful Degradation This approach attempts to modify a system so as to degrade its performance, assuming a particular hypothesis is correct. An example of this approach is pulling a spark plug wire on a running engine, in order to determine if that cylinder is firing. If the engine's performance degrades, then it was firing. This approach often involves removing redundancies in order to test a component in isolation. Unfortunately it is only applicable when the system is partially functional—that is, when there is room for further degradation of performance.

3.2.4 Evaluating Candidate Diagnoses

Some types of failures are more common than others. The likelihood or probability of a particular fault represents the expected frequency of that fault occurring in the given environment. Given a choice between two or more explanations of a faulty behavior, it is reasonable to prefer the one with the highest likelihood of occurring.

Probabilities may be associated with both the mechanisms and the resulting states of failure. Identifying the possible mechanisms of failure can influence the perceived likelihood of that failure mode. A mechanism may explain other suspect observations, or may be likely to occur given the pre-failure environment. On the other hand, all known mechanisms for a failure mode might be precluded by the pre-failure environment, making that hypothesis less credible. Associating possible mechanisms with failure modes is a difficult modeling issue requiring further research.

Multiple Faults Assuming independence allows us to compute joint probabilities for multiple faults, as the product of the individual probabilities for each fault. However, multiple faults are seldom independent; generally one primary fault occurs first, initiating a chain reaction of secondary faults. Identifying the mechanisms underlying this causal chain comprises an integral part of diagnosis, as it allows one to discard the independence assumption, and thus increase the credibility of the multiple-fault explanation. Finding explanations for how a primary fault could lead to secondary faults is similar to the problem of finding a mechanism explaining the primary fault. Causal models relating two faults are hard to come by but quite useful for evaluating multiple fault explanations.

3.2.5 Explanation in Diagnosis

In order for any reasoning system to be generally useful, it must be capable of explaining its own behavior. A diagnostic reasoning system should be able to answer the following kinds of questions:

1. How does the final diagnosis account for the observed symptoms?

2. How or why did a particular failure occur?
3. Why was a particular measurement or other action requested?
4. Why was one candidate hypothesis chosen over another?
5. What led the system to suspect a particular fault?
6. Why was a particular line of reasoning abandoned?

Answering these kinds of questions requires explicitly representing the reasoning steps leading to the final diagnosis. I believe that having such explicit representations should prove useful to more than just explanation—it should help control the diagnostic process as well. For example, the search for a valid model of a failed system could extend through the space of *diagnostic strategies* as well as the space of models.

4 Implementation

The implementation of a diagnostic engine is just getting started. However, several necessary components have been completed or are well underway. A variation of deKleer's assumption-based truth maintenance system, or ATMS, has been developed, offering some unique capabilities. A rule engine provides the interface to the ATMS. An incremental qualitative envisioner is nearly completed. These components have general utility in other research domains, and were developed in collaboration with Dennis DeCoste. These systems are described in the following pages.

The diagnostic engine, called PDE (for Process-based Diagnostic Engine) is mostly a collection of ideas at this point. I outline how I plan to implement it, using the other components described in this section. Figure 2 depicts the various modules and their interactions.

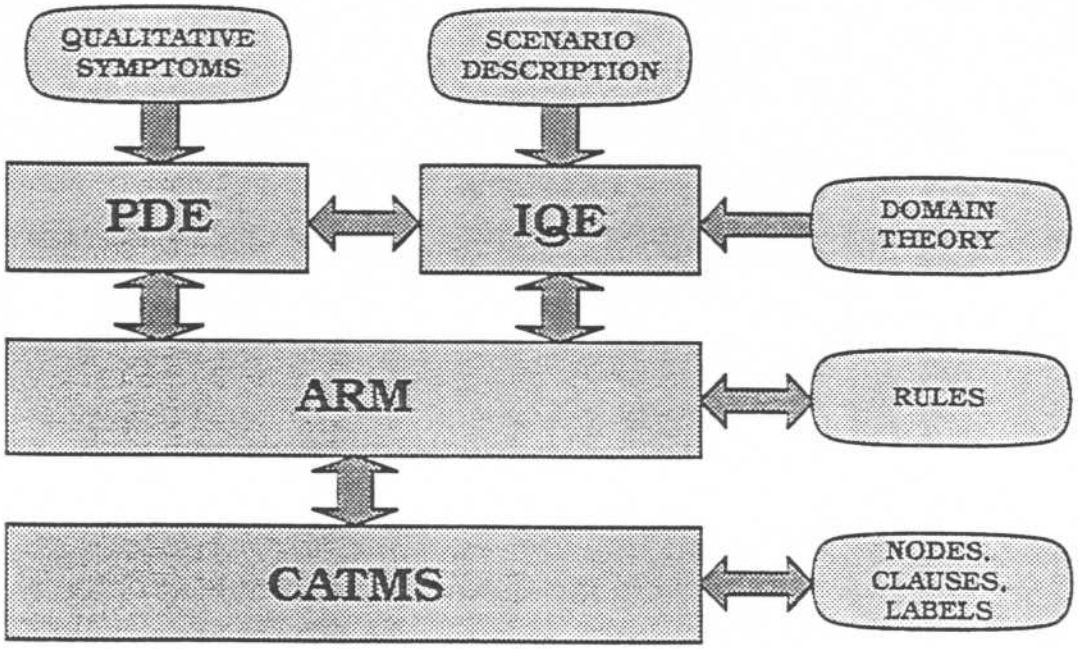
4.1 Assumption-Based Truth Maintenance

Given the view of diagnosis as a search through the space of models, it is essential that we have some way of representing and manipulating multiple models efficiently. It is natural to view a candidate model as a set of assumptions, where each assumption represents the presence of some component or module.

Assumption-based truth maintenance systems (ATMS) provide an efficient mechanism for reasoning with multiple situations simultaneously, thereby avoiding duplication of effort inherent in most backtracking schemes.

This section describes an approach for maintaining and reasoning with assumptions, based on a modified ATMS algorithm (CATMS) which uses *compressed labels* to represent the possible worlds in which propositions are believed. The CATMS algorithm is highlighted following a brief review of deKleer's original ATMS.

Figure 2: Diagnostic System Diagram



4.1.1 ATMS Basics

Propositions are represented by ATMS *nodes*. A subset of the nodes are designated *assumptions* by the problem solver. Sets of assumptions are represented by *environments*, which are used to represent possible worlds (eg., slices in time).

Logical relations among propositions are represented by *clauses* among nodes. A particular implementation of ATMS may support Horn clauses (justifications), CNF clauses, or both. Clauses have the effect of propagating truth between related nodes, starting with the nodes which have been assumed or asserted to be true.

An ATMS node may be true, false or unconstrained *in any given environment*. Rather than caching the deductive closure of every (interesting) environment, an ATMS associates with each node a *label*, indicating the set of environments in which the node is believed. The label of a node does not explicitly list all the environments in which it is true; instead, an ATMS uses a concise representation made possible by the property of monotonicity:

Definition 1 (Monotonicity Property) *If a node is true in a given environment $\mathcal{E}$, then it must also be true in every (consistent) environment containing $\mathcal{E}$ as a subset.*

The label of a node lists only those environments which (together with the clauses C) minimally entail the node. In other words, the label represents the set of most general environments in which the node must hold. Any consistent superset of some environment in a label (together with C) necessarily entails the node, due to the monotonicity property. This is what allows a minimal label to encode a node's environments, thereby avoiding the explicit mention of an exponentially large number of environments. The actual set of environments in which a node must hold may be viewed as a *version space*, which is bounded below by the node's label and above by the set of contradictory environments.

An environment is contradictory, or *nogood*, if it entails both N and $\neg N$ for some node N. Minimal nogoods arise from the "collision" of labels for N and $\neg N$, whenever one of these labels grows. A non-minimal nogood results when a minimal nogood subsumes some existing environment. Both types of nogoods must be removed from all labels in order to maintain consistency.

4.1.2 Introducing CATMS

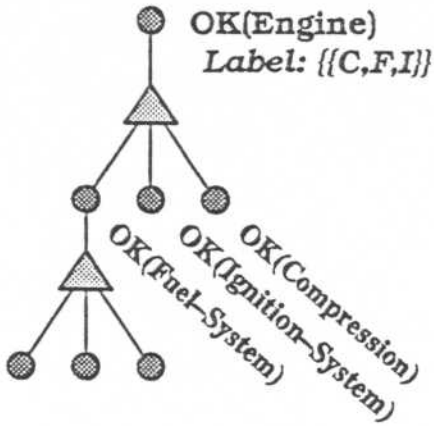
CATMS is a modified algorithm for Assumption-based Truth Maintenance. The "C" in CATMS stands for *compressed*, representing the fact that CATMS uses a compressed form of the node labels of a standard ATMS. CATMS differs from a standard ATMS in that the notion of environment is expanded to include the *closure* of the assumptions which it entails. That is, if other assumptions logically follow from the base assumptions of an environment and the clauses C, then those assumptions are considered to be part of that environment. The consequences of this simple modification are revealed below.

As defined above, minimality prevents a label from containing two environments such that one is a subset of the other. If the environments in a label are replaced by their assumption-closures, then minimality may eliminate certain environments which would be valid in a standard ATMS. This has the effect of *compressing* labels in CATMS. A CATMS label is always a subset of the corresponding standard ATMS label. In the case of an assumption, the label is always a single environment consisting of the assumption itself. Thus in the extreme case where all nodes are assumed, there is no label updating in CATMS, since every label will contain exactly one environment. If no assumptions receive justifications, then the labels in CATMS correspond identically to those in a standard ATMS.

Because CATMS labels are compressed, a query environment $\mathcal{E}$ must be *expanded* to include any implied assumptions before comparing it to environments in the node's label. The important observation is that by using assumption closures for subset tests during queries, the compressed label retains completeness; that is, the node is true in $\mathcal{E}$ if and only if there is an environment in its compressed label which is contained in the assumption-closure of $\mathcal{E}$.

This need to expand an environment before using it is offset by the smaller labels and overall

Figure 3: Hierarchical Representation of an Engine



fewer environments which must be explicitly represented. But CATMS provides more than just increased efficiency. Its compressed labels can be used to advantage, as shown below.

Hierarchical Representation Reasoning in general often proceeds in a hierarchical manner, starting with the most abstract level and gradually delving into the details. In diagnosis, we may wish to view a particular subsystem as a "black box" while reasoning about the system as a whole. Later as we begin to suspect that the subsystem is misbehaving, we may want to examine its internals in greater detail.

The compressed labels in CATMS provide a mechanism for just such hierarchical reasoning. Consider the example in Figure 3. The total system is believed to be functional when all of its major subsystems are operating properly. Each subsystem in turn is functional when all of its components are faultless. This decomposition may be continued to arbitrary depth. By assuming the status of each subassembly, the resulting labels reflect this decomposition. In the standard ATMS the lowest level details would be propagated all the way up to the system level, swamping the labels with unwanted detail.

Conceptually one could use the justifications themselves to achieve the hierarchy; but justifications do not generally provide good explanations; they are often too fine grained, and do not correspond well with explanatory levels a person might provide. Assumptions in CATMS may be chosen at the appropriate levels to provide the best explanations with the fewest steps to "get to the bottom of things".

This ability to reason hierarchically is of great significance to diagnosis. GDE [7] is unable to reason at multiple levels of detail, because it is based on the standard ATMS. Using CATMS, a GDE-

like approach can be extended to allow hierarchical representations and the efficiency of reasoning which they afford. The implementation of this idea is discussed in Section 4.4.

Implementation The CATMS algorithm described above has been implemented as a LISP program in a joint effort with Dennis DeCoste. The implementation has been tested and benchmarked on a number of examples, and in some cases significantly outperforms deKleer's ATMS—especially when assumptions are heavily justified. The CATMS implementation provides the knowledge base for all other components of the diagnostic system.

4.2 Assumption-based Rule Module: ARM

As with most truth maintenance systems, CATMS supports only propositional logic; variables are not supported. In order to make a TMS usable, a rule system provides the interface between it and the problem solver. Typically justifications in a TMS reflect the corresponding rules from which they were formed.

The rule module for CATMS is called ARM. It supports single and multiple-trigger rules, which may be conditioned to run immediately, or to wait until the triggers are believed. A focusing mechanism allows special rules to fire only when their antecedents hold under the current focus.

The body of a rule in ARM may be abitrary LISP code; however, the majority of rules simply install justifications in CATMS, where the antecedents of the justification correspond to the triggers of the rule.

Abductive Backchaining Every rule which installs a justification is cached to allow such rules to be run backwards in search of explanations. This style of reasoning is generally known as *abductive backchaining*, or *abduction*. Unlike deduction, abduction is not logically sound. It implicitly assumes that all phenomena have a *cause*; in addition, it requires a *closed-world assumption* that the known causes of some phenomenon are in fact the only possible ones. For example, if you walk outside and the grass is wet, then you might reasonably conclude that either it has rained or the sprinklers have been on recently. Explicit representation of the closed-world assumption allows such conclusions to be revised as other possibilities come to mind. This type of reasoning is very important to diagnosis, as will be shown in Section 4.4.

4.3 Incremental Qualitative Envisioning

In order to explore the space of qualitative models, we need the capability to reason efficiently with incremental changes to the original model. It is impractical to reason from scratch for each new variation of the evolving model. Existing qualitative simulators do not provide the needed flexibility for reasoning simultaneously with multiple models.

This section describes IQE (pronounced "IKE")—an Incremental Qualitative Envisioner being developed jointly with Dennis DeCoste. IQE is built on top of CATMS, our assumption-based truth maintenance system described in Section 4.1.2. The main goal of IQE is to provide a more flexible tool for analyzing qualitative models.

IQE is not intended to be a stand-alone program; instead it is designed as a component in a larger system, such as the diagnostic engine. IQE's flexibility is achieved by relinquishing many of the traditional duties of a qualitative simulator to the problem solver which calls it. Just as an ATMS maintains a space of environments, IQE maintains a *space of envisionments*. This allows for efficient exploration of the space of models, so that deviations from the initial structural description may be considered without starting from scratch.

IQE consists primarily of a translator which accepts qualitative domain theories (expressed in the language of QP theory) and converts these into ARM rules. In addition, IQE provides a set of rules implementing the constraints underlying qualitative reasoning. Other miscellaneous support facilities are provided, such as those needed for performing temporal reasoning.

IQE provides the foundation for reasoning qualitatively about a given scenario and domain theory. The domain theory is compiled into a set of rules in ARM, which is then fed a scenario description. The resulting justification structure in CATMS is then available for exploration.

4.3.1 Declarative Representation

One of the goals driving the development of IQE has been to develop and utilize a declarative knowledge base of the constraints underlying qualitative reasoning. In some cases this has proven impractical due to efficiency constraints—for example, transitivity is enforced using special purpose code which avoids the introduction of uninteresting inequality relations. Still, most of the reasoning in IQE is implemented by a single file of antecedent rules. These rules enforce various inequality constraints, resolve influences on quantities, and perform temporal reasoning. The rules are set up to infer consequences given the minimal required antecedents, and do not rely on complete state descriptions.

4.3.2 Representing Inequalities

Inequalities are central to qualitative reasoning in IQE. An inequality exists between two numbers whose ordering is relevant to the reasoning process. In IQE, a number may be either a quantity Q or its time derivative (D Q). IQE does not distinguish between variable and constant quantities; in particular, ZERO is treated as an ordinary quantity. Inequalities are limited to involve either two quantities or two derivatives; mixed inequalities are not allowed. This greatly simplifies the rules for reasoning about inequalities—particularly rules for temporal reasoning.

Inequalities are represented in IQE using a single relation: $\geq$. This representation scheme was chosen for its simplicity and expressive power. It avoids the canonical ordering problem of other

schemes, since both orderings of two related numbers are represented. It simplifies the rules used to implement inequality reasoning, since there are fewer cases to handle. It also reduces the number of nodes needed per inequality from four pairs (in QPE) down to two pairs. Finally, the justifications required to enforce taxonomy among the nodes are replaced by a single clause, requiring $\geq$ to be true for one of the two possible orderings of two related numbers, whenever both numbers exist.

The following table demonstrates how this scheme represents the possible inequality relations:

Relation	Representation using $\geq$
$Q_1 > Q_2$	$Q_1 \geq Q_2 \ \wedge \ Q_2 \not\geq Q_1$
$Q_1 = Q_2$	$Q_1 \geq Q_2 \ \wedge \ Q_2 \geq Q_1$
$Q_1 < Q_2$	$Q_1 \not\geq Q_2 \ \wedge \ Q_2 \geq Q_1$
$Q_1 \geq Q_2$	$Q_1 \geq Q_2$
$Q_1 \leq Q_2$	$Q_2 \geq Q_1$
$Q_1 \oslash Q_2$	$Q_1 \not\geq Q_2 \ \wedge \ Q_2 \not\geq Q_1$

4.3.3 Assumptions in IQE

In the course of building up a qualitative model, IQE posits four types of assumptions:

Scenario assumptions describe the structure of the given scenario. These are assumed rather than asserted to allow the problem solver (i.e., the diagnostic engine) to hypothesize structural deviations, perhaps to explain an observed spurious behavior.

Modeling assumptions represent the approximations and perspectives underlying the domain theory. Modeling assumptions provide the flexibility in perspective required for robustness over a broad domain. These may include *fault assumptions*, representing the belief that a particular type of fault exists, or *no-fault assumptions* indicating that a component or subsystem is in fact operating as intended.

State assumptions encode the qualitative state of the system, in terms of the active views and processes, or equivalently, the inequalities on which they are conditioned. These assumptions indicate the current mode of operation of the system.

Closed-world assumptions represent the belief that the individuals currently being considered are in fact the only ones. By explicitly assuming these facts, the space of their possible values may be searched for a match between observations and expectations.

4.3.4 Strategies for Manipulating Assumptions

A primary motivation for the development of IQE has been to avoid the combinatorics inherent in *interpretation construction*—the process of computing all maximal combinations of assumptions. There are two basic strategies for doing this. The first is to use *base assumptions* to focus the interpretation construction process as much as possible. The second is to avoid interpretation

construction altogether, and instead rely on the label computing mechanism of CATMS to produce the desired result. Each of these approaches has its strengths and shortcomings, as shown below.

Suppose we are interested in the steady state behavior of a fluid system. We could assert that all derivatives are zero, thus eliminating from consideration those states which imply change. Unfortunately, there is no easy way to retract a hard assertion in an ATMS, so we could never again reason about change without starting from scratch. One alternative which an ATMS makes possible is to *assume* the desired behavior (eg. steady state), and then build upon this *base assumption* when constructing interpretations. Base assumptions provide a focus to the interpretation construction process, without eliminating the option of changing focus later.

A second strategy is to configure the ATMS to compute the desired state(s) as part of its label update process. A *behavior node* is created to represent the desired behavior, and justified accordingly. Continuing the example above, a steady state node is justified by the conjunction of all quantities being constant. The label of this node will contain the set of minimal contexts in which all quantities are known to be constant.

The results obtained from these two approaches will generally differ, for the following reason. Interpretation construction finds all *maximal* contexts which are *consistent* with the given constraints, while the label approach finds all *minimal* contexts which *entail* the constraints. Interpretation construction has the advantage of completeness, while label computation assures that only relevant assumptions are considered.

For example, suppose there were in fact two independent fluid systems in the model, and we were only interested in the steady state behavior of one of them. Interpretation construction would combine the steady state behavior(s) of the system of interest with all possible behaviors of the other system. The label approach would never consider the second system, since it does not take part in the justification structure leading to the behavior node.

4.3.5 Incremental Temporal Reasoning in IQE

Qualitative reasoning derives much of its power from a few temporal constraints. Change is assumed to be continuous, and must be consistent with known temporal derivatives. Unlike other qualitative simulators which compute transitions between complete states, IQE performs temporal reasoning incrementally, requiring only partial state descriptions. Table 1 summarizes the temporal constraints for two temporally contiguous states, as enforced by IQE.

Temporal reasoning becomes important in diagnosis when symptoms occur *across time*. For example, there may be observations of the failure in progress, or the post-failure symptoms may involve oscillations or other non-steady behaviors. Dealing with dynamic behaviors is a particularly challenging aspect of this research.

Table 1: Constraints for Incremental Temporal Reasoning

Previous State		Next State	Clause
$Q_1 > Q_2$	$\Rightarrow$	$Q_1 \not< Q_2$	$Q_2 \geq Q_1 \vee \mathtt{Next}(Q_1 \geq Q_2)$
$Q_1 \not< Q_2 \wedge \dot{Q}_1 > \dot{Q}_2$	$\Rightarrow$	$Q_1 > Q_2$	$Q_1 \not\geq Q_2 \vee \dot{Q}_2 \geq \dot{Q}_1 \vee \mathtt{Next}(Q_2 \not\geq Q_1)$
$Q_1 > Q_2$	$\Leftarrow$	$Q_1 \not< Q_2 \wedge \dot{Q}_1 < \dot{Q}_2$	$Q_2 \not\geq Q_1 \vee \mathtt{Next}(Q_1 \not\geq Q_2) \vee \mathtt{Next}(\dot{Q}_1 \geq \dot{Q}_2)$
$Q_1 > Q_2 \wedge \dot{Q}_1 \not< \dot{Q}_2$	$\Rightarrow$	$Q_1 > Q_2$	$Q_2 \geq Q_1 \vee \dot{Q}_1 \not\geq \dot{Q}_2 \vee \mathtt{Next}(Q_2 \not\geq Q_1)$
$Q_1 > Q_2$	$\Leftarrow$	$Q_1 > Q_2 \wedge \dot{Q}_1 \not> \dot{Q}_2$	$Q_2 \not\geq Q_1 \vee \mathtt{Next}(Q_2 \geq Q_1) \vee \mathtt{Next}(\dot{Q}_2 \not\geq \dot{Q}_1)$
$Q_1 = Q_2$	$\Rightarrow$	$\mathtt{Diverging}(Q_1, Q_2)$	$Q_1 \not\geq Q_2 \vee Q_2 \not\geq Q_1 \vee \mathtt{Next}(\mathtt{Diverging}(Q_1, Q_2))$
$\mathtt{Converging}(Q_1, Q_2)$	$\Leftarrow$	$Q_1 = Q_2$	$\mathtt{Next}(Q_1 \not\geq Q_2) \vee \mathtt{Next}(Q_2 \not\geq Q_1) \vee \mathtt{Converging}(Q_1, Q_2)$
Interval	$\Leftrightarrow$	$\neg$Interval	$\neg\mathtt{Interval} \vee \mathtt{Next}(\neg\mathtt{Interval}), \mathtt{Interval} \vee \mathtt{Next}(\mathtt{Interval})$
$Q_1 > Q_2 \wedge \neg$Interval	$\Rightarrow$	$Q_1 > Q_2$	$Q_2 \geq Q_1 \vee \mathtt{Interval} \vee \mathtt{Next}(Q_2 \not\geq Q_1)$
$Q_1 > Q_2$	$\Leftarrow$	$Q_1 > Q_2 \wedge \neg\mathtt{Interval}$	$Q_2 \not\geq Q_1 \vee \mathtt{Next}(Q_2 \geq Q_1) \vee \mathtt{Next}(\mathtt{Interval})$
$Q_1 = Q_2 \wedge$Interval	$\Rightarrow$	$Q_1 \not> Q_2$	$Q_1 \not\geq Q_2 \vee Q_2 \not\geq Q_1 \vee \neg\mathtt{Interval} \vee \mathtt{Next}(Q_2 \geq Q_1)$
$Q_1 \not> Q_2$	$\Leftarrow$	$Q_1 = Q_2 \wedge$Interval	$Q_2 \geq Q_1 \vee \mathtt{Next}(Q_1 \geq Q_2) \vee \mathtt{Next}(Q_2 \geq Q_1) \vee \mathtt{Next}(\neg\mathtt{Interval})$

4.4 Process-Centered Diagnostic Engine: PDE

The Process-based Diagnostic Engine (PDE) is still under development. This section outlines its planned implementation, utilizing the other reasoning components already described.

The diagnostic engine is the top-level module in the overall system. It controls the other modules and provides all input and output functions. I assume as input a structural description of a system in its pre-failure condition, and a set of qualitative symptoms, as described in Section 3.2.1. In addition, I assume the availability of qualitative domain theory covering the given scenario.

The domain theory and scenario description are given to IQE, which translates the domain theory into a set of rules in ARM, and then feeds the scenario description to these rules. The result is a set of nodes and justifications in CATMS representing the qualitative constraints of the domain theory as they apply to the given scenario. This justification structure and the resulting node labels provide the foundation for all subsequent reasoning.

The diagnostic engine implements the AI paradigm of *generate and test*. Candidate hypotheses are generated and then tested against the domain theory and available observations. Additional measurements or other actions may be requested as a way of guiding hypothesis generation. Part of testing involves comparing alternative candidates to choose the best one. The search does not necessarily end with the first successful candidate, nor does it exhaustively enumerate all the possibilities. Some acceptibility criterion, based on the quality and likelihood of the explanation, must serve as the termination condition for search.

4.4.1 Generating Candidate Diagnoses

The first step in diagnosing the failure is to understand the nature of the conflict between the current model and the given symptoms. Specifically, the system must identify those aspects of the model leading to the failed expectations. This immediately removes from consideration all portions of the model which continue to yield valid predictions of behavior.

Finding Predictive Models In a few fortunate cases, a simple variation of the original model may actually predict the observed symptoms. Changing a modeling assumption or activating an explicit fault assumption may be all that is required to make the discrepancies disappear. This reduces to finding one or more fault assumptions which together imply the observed behavior. This can be done in CATMS by creating a node representing the faulty behavior and examining its label. This *behavior node* represents the conjunction of the individual behavioral observations, and is justified accordingly. Its label consists of the minimal combinations of faults capable of explaining the behavior.

This approach has its origins in a similar technique introduced by Falkenhainer and Forbus [10] as a means of focusing query answering. Note that this technique works equally well for single and multiple faults. Single faults show up as singleton contexts in the node's label. It is limited by its dependence on explicit fault models, which will not always be available.

Finding Consistent Models In many cases, no explicit model of the specific fault behavior is available. This situation is indicated by an empty label for the behavior node. That is, no combination of fault assumptions suffices to account for the observations. Having failed to find a predictive model, we settle for a model which is *consistent* with observed behavior. Of course, the *null model*, consisting of no structural constraints at all, will always be consistent with any behavior. But we are interested in *minimal* deviations from the original model which are consistent with observations.

deKleer's General Diagnostic Engine (GDE) [7] provides a mechanism for finding maximal assumption sets consistent with observed behavior. Each assumption indicates the correct behavior of a single component. GDE uses an ATMS to find the minimal contexts in which the model's predictions conflict with actual observed behavior. These conflicting contexts can be found by examining the label of a *conflict node*, which is just the negation of the behavior node discussed previously. The set of possible *culprits* is obtained by choosing one assumption from each conflicting context and combining them, in all possible ways. Non-minimal culprits are discarded. The remaining culprits represent minimal sets of components whose failure is consistent with observations.

Exploring Consistent Models I plan to borrow the above technique, but use it in a novel way. In GDE, an assumption always represents the presence of a functioning component. The hierarchical

nature of CATMS allows similar assumptions about the workings of entire subsystems. If a subsystem is identified as a possible culprit, then it may be recursively examined down to the component level if desired, in search of an explanation of the observed symptomatic behavior. Another difference in this approach is that there are modeling assumptions and closed world assumptions which can show up as culprits. Violation of a modeling assumption suggests that a change in perspective might yield consistent predictions. By allowing a closed world assumption to be violated, the system can hypothesize active processes which were previously unknown (or missing processes previously believed to be active), by backchaining through the rules of the domain theory.

4.4.2 Testing Candidate Diagnoses

A Candidate diagnosis generated by the above approach is not guaranteed to be valid. The approach may yield multiple candidates, at most one of which is correct. Each candidate must be analyzed to verify that it actually explains the anomolous behavior. This is done by simulating the altered model suggested by the candidate, to see if the observed symptomatic behavior is among those predicted by the qualitative simulation.

Candidate Selection A candidate diagnosis may explain the anomolous behavior and still not be the correct diagnosis. When multiple diagnoses survive the verification step above, the candidates must be ordered based on their relative likelihoods. The likelihood of a fault will be estimated based on knowledge of the reliability of the failed component, or on the likelihood of the believed mechanism of failure. The exact manner in which these are to be calculated remains to be determined.

Active Diagnosis Multiple diagnoses may also be resolved by obtaining additional information about the failed system. Since I will not have access to a physical prototype of the systems I plan to model, I will simulate the actions suggested by the diagnostic engine and return their consequences. Defining the possible actions which may be performed will likely require a formal *language of actions*, as well as a good deal of modeling. This work is underway.

5 An Example

There are still many issues to be resolved and the implementation is far from complete, so a detailed example is not possible; however, walking through a simple example may help to clarify the basic approach.

Consider a fuel leak in a turbofan aircraft; pressure sensors indicate nominal pressure at the fuel pump, but sub-nominal fuel pressure at the starboard engine. Engine RPMs and thrust are also lower than expected.

Diagnosing this situation would proceed as follows. If the domain theory includes a fault model for a leaky fluid path, then the corresponding fault assumption is propagated and combined with relevant scenario assumptions to predict the observed behavior—there is a non-empty label for the behavior node representing the symptomatic behavior. Similarly, fault models for the pressure sensors and other aspects of the engine's performance may also combine to explain the symptoms.

In the event that no explicit fault model is available, then maximal consistent subsets of the original model are sought, by identifying minimal culprits and retracting them. In this case, the fuel subsystem will likely be the only minimal culprit, so retracting the assumption that the fuel system is operational will regain consistency. The fuel system is then examined in isolation; that is, the "black box" is opened up. Eventually this process focusses in on the level of individual components, such as the failed fuel line.

Suspending the constraints associated with the fuel line produces a consistent model, but does not explain the symptoms; doing so requires replacing the fuel line (in the model, not the aircraft) with whatever it has become. This is where abduction plays a role. The domain theory is searched for a process structure consistent with the fuel line's behavior. Each candidate process structure is then instantiated by filling roles with posited individuals. One such process structure in this case would have an additional fluid flow process from the fuel line to some unknown location. Additional knowledge of spatial continuity could limit the possibilities to those adjacent to the fuel line. The resulting model is analyzed to verify that the hypothesized failure can account for the observed behavior of the fuel line.

6 Summary

This research investigates the use of qualitative reasoning in diagnosis. The approach taken is to view diagnosis as a problem of minimally perturbing a faulty model of a system until it matches the observed behavior of that system. This differs from the more traditional view that the fault lies in the system itself. By recognizing that the system continues to operate "correctly" as viewed by the laws of nature, we may apply a single domain theory both to derive the model and to repair it.

A model may be faulty in different ways. It may be that the structural description of the scenario is inaccurate. Simplifying assumptions may have been made inappropriately. Or the theory underlying the model may be fundamentally flawed. This research focuses on identifying faults in scenario descriptions and simplifying assumptions.

I claim that qualitative models can account for observed symptoms without relying on explicit fault models, thereby increasing robustness for novel faults. Both the mechanism of failure and the resulting state of the failed system can be modeled in terms of their underlying active processes. By focusing on the process structure rather than the physical structure, the search space is both

smaller and more focused, since only those processes which could influence the system in the observed directions need to be considered.

A variety of methods may be employed in the search for a valid model. The simplest is to look for variants of the model which predict or explain the anomalous behavior directly. These will likely contain nonstandard modeling assumptions or explicit fault assumptions. If this fails, the search may extend to variants which are *consistent* with observations. This is guaranteed to succeed, since removing all structural constraints would necessarily eliminate the inconsistency.

Candidates must be evaluated and compared to determine the most likely diagnosis. Evaluation is based on the number and nature of the deviations from the original model. By hypothesizing a mechanism by which the original system is transformed into its faulted form, the evaluation is reduced to establishing the likelihood of the transformation. For example, a blown fuse provides a more likely explanation if we know how the fuse might have become blown. Qualitative reasoning can both provide focus on a suspected failure and explain how and why it may have occurred.

Other diagnostic strategies to be investigated include measurement planning and its generalization: *active diagnosis*, whereby a system is modified to test a working hypothesis. An example of active diagnosis is pulling a spark plug wire on a poorly-running engine to determine if a cylinder is misfiring. My goal is to define a theory of interaction, including a language for representing possible actions and their expected consequences.

This approach to diagnosis will be tested on a number of systems and fault types. An implementation is in progress, building on a modified ATMS and an incremental qualitative envisioner. So far I have chosen four test systems: a ship's boiler; a thermal control system for the US space station; a gasoline piston engine and a jet turbine engine. The tests will hopefully include dynamic behaviors, using temporal reasoning techniques being developed jointly with Dennis DeCoste. Specific faults for each system will be chosen as the research progresses.

7 Acknowledgements

Dennis DeCoste and Gordon Skorstad have contributed significantly to this research. This research is supported by the National Aeronautics and Space Administration, Contract No. NASA NAG 11023.

References

[1] Allen, J. "Towards a general model of action and time" *Artificial Intelligence*, 23(2), July 1984.

[2] Collins, J. and Forbus, K. "Building qualitative models of thermodynamic processes", *submitted for publication, 1990*

[3] Collins, J. and Forbus, K. "Reasoning about Fluids via Molecular Collections", in *Proceedings of the National Conference on Artificial Intelligence*, Seattle, July, 1987.

[4] Davis, R. et al. "Diagnosis based on Structure and Behavior", in *Proceedings of the National Conference on Artificial Intelligence*, August, 1982.

[5] Davis, R. "Diagnostic Reasoning based on Structure and Behavior", *Artificial Intelligence*, **24**, 1984.

[6] de Kleer, J. "An Assumption-Based Truth Maintenance System", *Artificial Intelligence*, **28**, 1986.

[7] de Kleer, J. and Williams, B. "Diagnosing Multiple Faults", *Artificial Intelligence*, **32**, 1987.

[8] de Kleer, J. and Brown, J. "A Qualitative Physics based on Confluences", *Artificial Intelligence*, **24**, 1984.

[9] de Kleer, J. and Bobrow, D. "Qualitative Reasoning with Higher Order Derivatives", in *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August, 1984.

[10] Falkenhainer, B. and Forbus, K. "Setting up large-scale qualitative models", *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306, St. Paul, MN, August 1988. Morgan Kaufmann.

[11] Forbus, K. "Qualitative Process Theory" *Artificial Intelligence*, **24**, 1984.

[12] Forbus, K. "The Logic of Occurrence", Proceedings of IJCAI-87, August, 1987.

[13] Forbus, K. "The Qualitative Process Engine" in *Readings in Qualitative Reasoning about Physical Systems*, Weld, D. and de Kleer, J. (Eds.), Morgan Kaufmann, 1990.

[14] Genesereth, M. "Diagnosis using Hierarchical Design Models", *Proceedings of the National Conference on Artificial Intelligence*, 1982.

[15] Hayes, P. "The Naive Physics Manifesto", in *Expert systems in the Micro-Electronic Age*, D. Michie (Ed.), Edinburgh University Press, 1979.

[16] Hayes, P. "Naive Physics 1: Ontology for Liquids", in Hobbs, J. and Moore, B. (Eds.), *Formal Theories of the Commonsense World*, Ablex Publishing Corporation, 1985.

[17] Haywood, R. W. *Analysis of Engineering Cycles*, Pergamon Press, 1980.

[18] Iwasaki, Y., and Simon, H. "Causality in Device Behavior", *Artificial Intelligence*, **29**, 1986.

[19] Kuipers, B. "Common Sense Causality: Deriving Behavior from Structure", *Artificial Intelligence*, **24**, 1984.

[20] Kuipers, B. "Abstraction by Time-Scale in Qualitative Simulation", in *Proceedings of the National Conference on Artificial Intelligence*, Seattle, July, 1987.

[21] Mohammed, J., and Simmons, R. "Qualitative simulation of semiconductor fabrication", Proceedings of AAAI-86, August, 1986.

[22] Reynolds, W. and Perkins, H. (1977) *Engineering Thermodynamics.* McGraw-Hill Press, New York, New York.

[23] Shortliffe, E. H. "Computer-Based Medical Consultations: MYCIN", New York, American Elsevier. 1976.

[24] Simmons, R. "Representing and reasoning about change in geologic interpretation", MIT Artificial Intelligence Lab TR-749, December, 1983.

[25] Skorstad, G. and Forbus, K. "Qualitative and quantitative reasoning about thermodynamics" *Proceedings of the eleventh annual conference of the Cognitive Science Society*, Ann Arbor, MI, August, 1989.

[26] Weld, D. "Comparative Analysis", Proceedings of IJCAI-87, August, 1987.

[27] Williams, B. "Qualitative Analysis of MOS Circuits", *Artificial Intelligence*, 24, 1984.

[28] Williams, B. "The Use of Continuity in a Qualitative Physics", in *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August, 1984.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-91-1699	2.	3. Recipient's Accession No.

4. Title and Subtitle

DIAGNOSIS AS FAILURE UNDERSTANDING

5. Report Date

6.

7. Author(s)

JOHN W. COLLINS

8. Performing Organization Rept. No. R-91-1699

9. Performing Organization Name and Address

Department of Computer Science
1304 W. Springfield
Urbana, IL 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

NASA NAG 11023

12. Sponsoring Organization Name and Address

NASA

13. Type of Report & Period Covered

Technical

14.

15. Supplementary Notes

16. Abstracts

This paper reports on research in progress, investigating the use of process-centered qualitative models in the diagnosis of continuous variable systems. I view diagnosis as a problem of minimally perturbing a faulty model of a system until it matches observed behavior. I claim that qualitative models can account for observed symptoms without relying on explicit fault models, thereby increasing robustness for novel faults. Both the mechanism of failure and the resulting state of the failed system can be modeled in terms of their underlying active processes. By focusing on the process structure rather than the physical structure, the search space is both smaller and more focused, since only those processes which could influence the system in the observed directions need to be considered. Several diagnostic strategies are being explored within this framework. An implementation is in progress, building on a modified ATMS and an incremental qualitative envisioner. The kinds of systems I am examining are taken from the domain of thermodynamics, and include piston and gas-turbine engines, a steam boiler and a thermal control system.

17. Key Words and Document Analysis. 17a. Descriptors

Diagnosis
Qualitative Reasoning

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
unlimited	UNCLASSIFIED	30
	20. Security Class (This Page) UNCLASSIFIED	22. Price