MONITORING AND DIAGNOSIS OF CONTINUOUS DYNAMIC SYSTEMS USING SEMIQUANTITATIVE SIMULATION

by

DANIEL LOUIS DVORAK, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May, 1992

Acknowledgments

I especially thank Prof. Benjamin Kuipers for his guidance in shaping this research and his continuous encouragement; many times his insightful comments helped me to press forward, and many times his encouraging words helped me believe that the work was worth doing. I am indebted to Dan Berleant for his work on Q2/Q3 and to Bert Kay for his work on dynamic envelopes; both of these research efforts provided important tools to build upon. I sincerely appreciate the efforts of Kee Kimbrell and Dan Clancy in porting QSIM to the Macintosh. I value the many other friendships that I gained during this time, including those of Ray Bareiss, Jimi Crawford, Adam Farquhar, David Franke, Wan-Yik Lee, Wood Lee, Raman Rajagopalan, David Throop, and Chris Walton.

I sincerely appreciate the encouragement and support of Reid Watts and Harold Jackson of AT&T Bell Laboratories, and the financial assistance of the Doctoral Support Program at AT&T Bell Laboratories. I thank the Visionnaire Group at Bell Labs for the use of their Symbolics machines.

Finally, I thank my wife Wafá for her patience and understanding during the last two years as I toiled into the night.

Daniel Louis Dvorak

The University of Texas at Austin
May, 1992

MONITORING AND DIAGNOSIS OF CONTINUOUS DYNAMIC SYSTEMS USING SEMIQUANTITATIVE SIMULATION

Publication No.______________

Daniel Louis Dvorak, Ph.D.
The University of Texas at Austin, 1992

Supervisor: Benjamin J. Kuipers

Operative diagnosis, or diagnosis of a physical system in operation, is essential for systems that cannot be stopped every time an anomaly is detected, such as in the process industries, space missions, and medicine. Compared to maintenance diagnosis where the system is off-line and arbitrary points can be probed, operative diagnosis is limited mainly to sensor readings, and diagnosis begins while the effects of a fault are still propagating. Symptoms change as the system's dynamic behavior unfolds.

This research presents a design for monitoring and diagnosis of deterministic continuous dynamic systems based on the paradigms of "monitoring as model corroboration" and "diagnosis as model modification" in which a semiquantitative model of a physical system is simulated in synchrony with incoming sensor readings. When sensor readings disagree with predictions, variant models are created representing different fault hypotheses. These models are then simulated and either corroborated or refuted as new readings arrive. The set of models changes as new hypotheses are generated and as old hypotheses are exonerated. In contrast to methods that base diagnosis on a snapshot of behavior, this

simulation-based approach exploits the system's time-varying behavior for diagnostic clues and exploits the predictive power of the model to forewarn of imminent hazards.

The design holds several other advantages over existing methods: 1) semiquantitative models provide greater expressive power for states of incomplete knowledge than differential equations, thus eliminating certain modeling compromises; 2) semiquantitative simulation generates guaranteed bounds on variables, thus providing dynamic alarm thresholds and thus fewer fault detection errors than with fixed-threshold alarms; 3) the guaranteed prediction of all valid behaviors eliminates the "missing prediction bug" in diagnosis; 4) the branching-time description of behavior permits recognition of all valid manifestations of a fault (and of interacting faults); 5) hypotheses based on predictive semiquantitative models are more informative because they show the values of unseen variables and can predict future consequences; and 6) fault detection degrades gracefully as multiple faults are diagnosed over time.

Table of Contents

Acknowledgments **ii**

Abstract **iii**

Table of Contents **v**

1. Introduction **1**

1.1 The Problem: Monitoring Dynamic Systems 1

1.1.1 Background and Motivation 1

1.1.2 Operative Diagnosis 2

1.1.3 Operator Advisory Systems 4

1.1.4 False Positives, False Negatives 5

1.1.5 The Domain 6

1.2 The Approach 7

1.2.1 Goals and Non-Goals 7

1.2.2 Diagnosis as Model Modification 8

1.2.3 Semiquantitative Simulation 8

1.2.4 Benefits 9

1.2.5 Architecture 10

1.3 Scope 10

1.3.1 Assumptions 10

1.3.2 Non-requirements 11

1.3.3 Modeling Issues 11

1.3.4 Implementation 12

1.3.5 Empirical Evaluation 13

1.4 Claims 13

1.4.1 Modeling & Simulation 13
1.4.2 Predictive Monitoring 14
1.4.3 Discrepancy Detection & Diagnosis 14
1.4.4 Skepticism 15
1.5 Example: A Two-Tank Cascade 17
1.5.1 Modeling 17
1.5.2 Simulation 18
1.5.3 Discrepancy Detection 20
1.5.4 Hypothesis Generation 21
1.5.5 Hypothesis Testing 23
1.5.6 Forewarning 24
1.5.7 Summary 24
1.6 Guide to the Dissertation 25
1.7 Terminology 25

2. Related Work **31**
2.1 Symptom-Based Approaches 31
2.1.1 Rule-Based Systems 31
2.1.2 Fault Dictionaries 33
2.1.3 Decision Trees 33
2.2 Model-Based Approaches 34
2.2.1 PREMON/SELMON (Doyle et al.) 34
2.2.2 DRAPHYS (Abbott) 36
2.2.3 MIDAS (Finch, Oyeleye and Kramer) 37
2.2.4 Inc-Diagnose (Ng) 39
2.2.5 Kalman Filters 41

2.2.6 KARDIO (Bratko et al.) . . . 45
2.2.7 Modeling for Troubleshooting (Hamscher) . . . 49
2.3 Influential Research . . . 51
2.3.1 Measurement Interpretation . . . 51
2.3.2 Generate, Test and Debug . . . 52
2.3.3 STEAMER . . . 54

3. The Design of Mimic **55**
3.1 Design Overview . . . 55
3.2 Modeling . . . 58
3.2.1 Structural Model . . . 58
3.2.2 Behavioral Model . . . 62
3.2.3 Modeling Faults . . . 64
3.3 Simulation . . . 65
3.3.1 Qualitative-Quantitative Simulation . . . 66
3.3.2 Feedback Loops . . . 67
3.3.3 State-Insertion for Measurements . . . 70
3.3.4 Dynamic Envelopes . . . 72
3.3.5 Pruned Envisionment . . . 73
3.4 Monitoring . . . 73
3.4.1 Monitoring Model . . . 74
3.4.2 Limitations of Alarms . . . 75
3.4.3 Discrepancy Detection . . . 76
3.4.4 Tracking . . . 79
3.4.5 Updating Predictions from Measurements . . . 81
3.4.6 Measurement Issues . . . 81

3.5 Diagnosis . . . 82
3.5.1 Hypothesis Generation . . . 83
3.5.2 Hypothesis Testing . . . 86
3.5.3 Resimulation . . . 88
3.5.4 Hypothesis Discrimination . . . 89
3.5.5 Multiple-Fault Diagnosis . . . 90
3.6 Advising . . . 90
3.6.1 Warning Predicates . . . 90
3.6.2 Forewarning . . . 90
3.6.3 Ranking of Hypotheses . . . 91
3.6.4 Defects vs. Disturbances . . . 92
3.7 Special Fault Handling . . . 93
3.7.1 Intermittent Faults . . . 93
3.7.2 Consequential Faults . . . 93
3.8 Controlling Complexity . . . 93
4. Experimental Results **97**
4.1 Gravity-Flow Tank . . . 97
4.1.1 Cycle 9: t = 0.9 . . . 99
4.1.2 Cycle 11: t = 1.1 . . . 102
4.1.3 Cycle 12: t = 1.2 . . . 103
4.1.4 Cycle 22: t = 2.2 . . . 104
4.2 Two-Tank Cascade . . . 104
4.3 Open-Ended U-Tube . . . 108
4.4 Vacuum Chamber . . . 111
4.5 The Dynamics Debate . . . 113

5. Discussion and Conclusions **115**

5.1 Design Principles 115

5.2 Strengths 117

5.3 Limitations 119

5.3.1 Temporal Abstraction 119

5.3.2 Dependency Tracing 119

5.3.3 Spurious Behaviors 120

5.3.4 Cascading Faults 120

5.3.5 Complexity and Real-Time Performance 121

5.4 Appropriate Domains 121

5.5 Future Work 122

5.5.1 Discrepancy Detection 122

5.5.2 Perturbation Analysis 122

5.5.3 Component-Connection Models 123

5.5.4 Hierarchical Representation and Diagnosis 123

5.5.5 Scale-Space Filtering 125

5.5.6 Speeding it Up 125

5.5.7 Reconciling FDI and MBR 126

5.5.8 Real Applications 126

5.6 Epilogue 127

A. Sample Execution History **129**

BIBLIOGRAPHY **149**

Vita

Chapter 1

Introduction

> *...computer technology allows more complex systems to be controlled but it also enables far more information about the system to be displayed. For example, process control and instrumentation systems such as those found in nuclear power plants may have around 2000 alarms in a control room in addition to the displays of analogue plant data. When plant crises occur, these data can change very rapidly. In one simulated loss-of-coolant accident, 500 lights went on or off within the first minute, and 800 in the second.*
>
> *It is in these kinds of real-time problem-solving situations that many of the limitations of humans are at their most apparent. Their tendency to overlook relevant information, to respond too slowly and to panic when the rate of information flow is too great all contribute to lower than desired levels of performance.* [SPT86]

1.1 The Problem: Monitoring Dynamic Systems

1.1.1 Background and Motivation

Nuclear power plant operations is but one example in which a human must monitor a physical system through sensor readings and interpret those readings, often under grave time pressure, when unexpected behavior occurs. This task is prevalent in our modern world — in operating a petroleum refinery, in flying a commercial jet aircraft, in monitoring a patient in a surgical intensive care unit, and in monitoring spacecraft environmental systems. In all these cases the human's primary source of information is sensor readings and alarms, and diagnosis must be performed while the system continues to operate.

The hazards of performing this difficult task incorrectly or too slowly can be seen in the records of notable accidents such as the 1979 Three Mile Island nuclear power accident, the 1977 New York City blackout, the near-tragedy of the *Apollo 13* flight in 1970, and the 1969 Texas City explosion of a butadiene refining unit. As Perrow recognized when he analyzed these and other accidents, the characteristics that make a system more prone to accident are *tight coupling* and *complex interactions* [Per84]. Such systems are harder to understand when something goes wrong and allow less time to take corrective action. Perrow's interactions/coupling chart (Figure 1.1) shows examples of high-risk systems in the

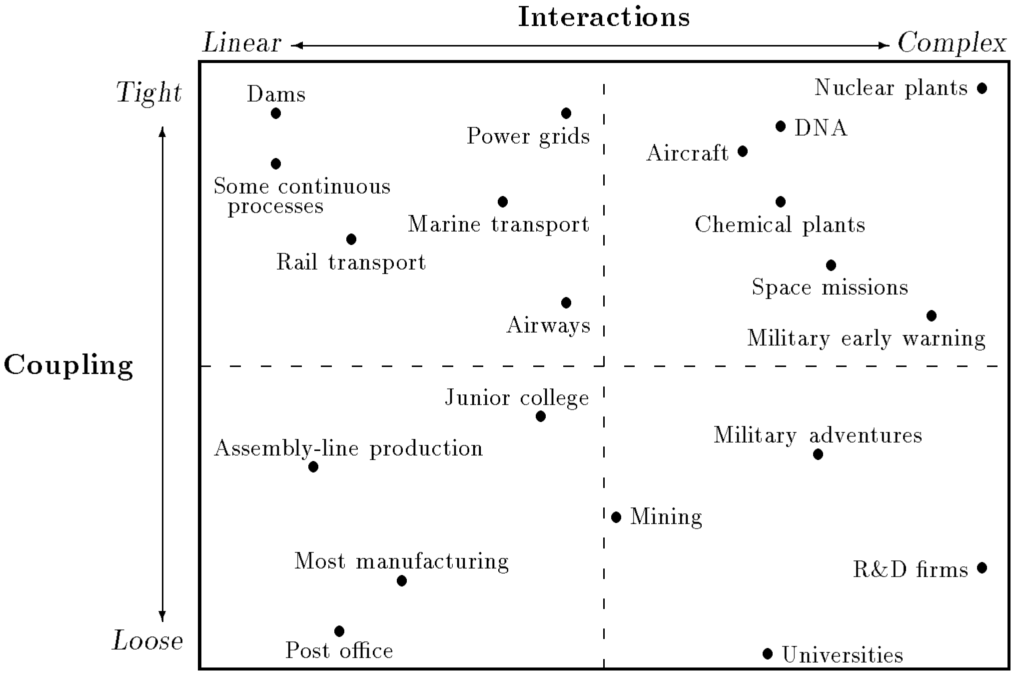


Figure 1.1: Perrow's Interactions/Coupling Chart. The systems most prone to accident are those having tight coupling and complex interactions, as shown in the upper-right quadrant.

upper right quadrant, such as nuclear plants, chemical plants, aircraft, and space missions. (Medical intensive care also belongs in this quadrant since human biology exhibits tight coupling and complex interactions; Perrow did not include medical care because his focus was on man-made organizations and technology.)

1.1.2 Operative Diagnosis

This research focuses on the problem of *operative diagnosis*, or diagnosis of physical systems in operation. As Abbott notes [Abb90, p. 135], operative diagnosis is distinct from "off-line" or "maintenance" diagnosis in several respects. As Table 1.1 shows, the objective of operative diagnosis is to facilitate continued safe operation of the system whereas the objective of maintenance diagnosis is to determine which part to fix or replace. Operative diagnosis arises in at least three situations: when it is impossible to shut down the physical system (as in medicine), when faults are tolerated because it is too expensive to stop for every maintenance item (in industry), and when severe consequences may result within seconds or minutes after a malfunction (as in space missions). Malkoff [Mal87, p. 98] describes well a root problem in operative diagnosis:

	Operative diagnosis	**Maintenance diagnosis**
Context	System remains in operation.	System is off-line.
Objective	Continued safe operation.	Fix or replace faulty component.
Requirements	Identify faulty component, identify specific fault, expose its effects, and forewarn of possible adverse effects.	Localize fault to a field replaceable unit.
Observations	Limited mainly to sensor readings.	Can probe arbitrary points.
Symptoms	Diagnosis begins while effects of fault are still propagating, so symptoms may change.	Diagnosis usually done after all effects have propagated.
Hypothesis	Consists of a mechanism model embodying zero or more faults, plus its state.	Identifies a suspected component.
Testing	Restricted to cautious input perturbations.	Can apply arbitrary input signals.

Table 1.1: Operative Diagnosis versus Maintenance Diagnosis. Much of the prior work in knowledge-based diagnosis is aimed at maintenance diagnosis. This report addresses the challenges of operative diagnosis.

In process control systems, the detection and diagnosis of faults generally depends upon two mechanisms:

1. Large numbers of sensors are installed at key points in the plant. Raw parameter values transmitted from these sensors are monitored and compared with pre-specified upper and lower range-limits of normal. When parameter range-limits are exceeded, alarms are activated to attract the attention of the operator.
2. Human operators are responsible for performing (manually, and in real time) the multisensor integration (i.e., the process of observing all the sensor data), particularly the occurrence of alarms, analyzing their significance and correctly formulating a diagnosis.

Herein lies the root of a serious problem in dealing with fault diagnosis:

> Design engineers and, to a lesser extent, plant operators, can reasonably well anticipate the pattern of alarms that will be triggered by a known malfunction. To a lesser degree, this is the case even for multiple, simultaneous malfunctions. On the other hand, they have great difficulty reasoning in the reverse direction: that is, mapping from a complicated pattern of sensor alarms back to causative faults.

1.1.3 Operator Advisory Systems

To help the operator perform operative diagnosis, the "operator advisory system" has emerged as an extension to existing monitoring & control technology (see Figure 1.2), and has become an important area of application for expert systems. ESCORT [SPT86] (an expert system for complex operations in real-time) and REALM [TC86] (a reactor emergency action level monitor) are two of many expert systems developed for process industries. (For surveys of this work, see Dvorak's study of monitoring & control expert systems [Dvo87] and Laffey et al.'s survey of real-time knowledge-based systems [LCS+88].) These systems aim to reduce the cognitive load on operators, usually by helping to diagnose the cause of alarms and possibly suggesting corrective actions. Most of these expert systems get their knowledge of symptoms, faults, and corrective actions through the usual process of codifying human expertise in rules or decision trees. But the problem, as with all expert systems, is reliability. As Denning observes,

> *the trial-and-error process by which knowledge is elicited, programmed, and tested is likely to produce inconsistent and incomplete databases; hence, an expert system may exhibit important gaps in knowledge at unexpected times* [Den86].

These "gaps in knowledge" can lead to errors in fault detection and diagnosis, and can have serious consequences in some applications.

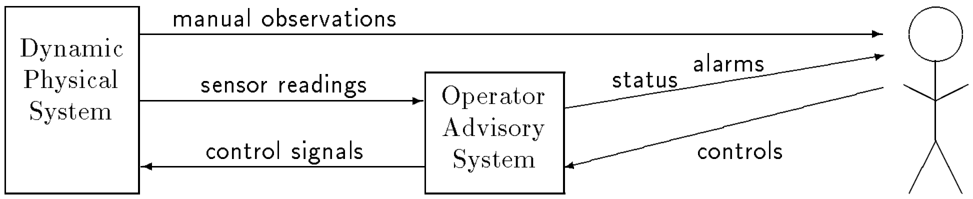


Figure 1.2: Operating a physical system. The purpose of the operator advisory system is to assist the operator in interpreting sensor readings and determining appropriate control actions.

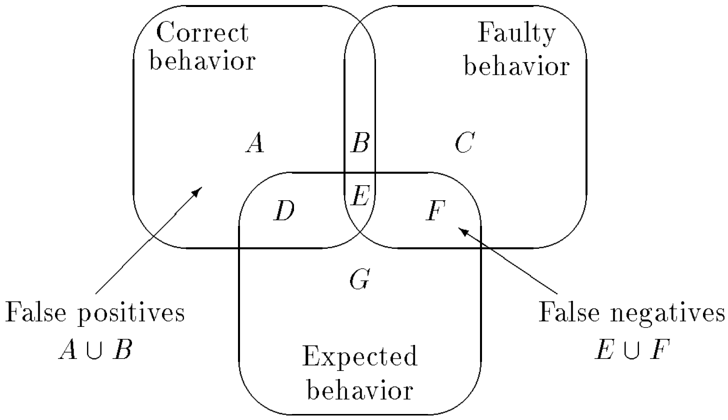


Figure 1.3: Fault detection errors as a function of behavior. False positives arise when expected behavior fails to include all valid correct behaviors. False negatives arise when faulty behavior is indistinguishable from expected behavior.

1.1.4 False Positives, False Negatives

Fault detection can be wrong in two ways: as *false negatives*, in which a real fault goes undetected, and as *false positives*, in which an alarm is raised when no fault is present (see Figure 1.3). There are several fundamental reasons why a fault may be undetectable, and thus cause a false negative: the fault may be masked by a redundant spare; the fault may not be exposed in the current operating mode (such as a burned-out light bulb with no power applied); the fault may not have affected any sensor yet, or the affected sensor may itself be defective; the fault manifestations may be buried in noise or may simply be too small to distinguish from normal behavior, particularly in its early stages.

Similarly, there are several reasons why a fault may be "detected" when none is present: readings from the fault-free mechanism may exceed a detection threshold, whether due to noise or to acceptable variations within the mechanism; thresholds designed for steady-state operation may be exceeded during other phases of operation, such as startup and shutdown; and normal-but-infrequent behavior may be neglected in threshold design. False positives (a.k.a. false alarms) might seem like a less serious problem since, in some critical applications, it is better to be safe than sorry. However, the following caution is given in a book on fault detection:

> *False alarms are generally indicative of poor performance in a fault detection scheme. Even a small false alarm rate during normal operation of the monitored system is unacceptable because it quickly leads to lack of confidence in the detection scheme.* [CFP89, p. 8]

The design of alarm thresholds is usually viewed as a tradeoff in which narrow thresholds cause false positives and wide thresholds cause false negatives. The objective is to strike a compromise where the rates of false positives and false negatives are acceptable for the given monitoring situation. However, this view lumps together all sources of uncertainty and assumes that limit-checking is the only way of detecting faults. Later in Chapter 3 we will show how it is possible to eliminate (in a mathematical sense) some of the sources of false positives.

This research has been motivated both by need and by opportunity. The need clearly exists for improved methods of monitoring and diagnosis of continuous dynamic systems; when something goes wrong in a complex system, the operator needs help in heeding all data and forming explanations that account for the data. Advances in the field of qualitative reasoning and semiquantitative reasoning have generated an opportunity to provide a new foundation for operator advisory systems that offers distinct improvements over current practice.

1.1.5 The Domain

The technology described in this dissertation is applicable to deterministic, continuous-variable dynamic systems that can be modeled, at least approximately, with ordinary differential equations. This encompasses many physical phenomena that are well understood through the laws of physics, such as thermodynamics, fluid mechanics, and electricity. However, a strength of this technology is that it enables modeling of real-world systems in which knowledge of parameter values and functional relationships is imprecise, and for which analytic models are not solvable. Such incomplete knowledge occurs not only in fields where our understanding is incomplete (such as in human physiology) but also in real-world mechanisms where device performance is expressed as being within a specified tolerance range (such as a water pump). A sampling of potential applications for this technology includes systems such as thermal power plants, chemical refineries, jet engines, intensive-care monitoring, and spacecraft environmental systems.

This research focuses on the problems of monitoring and diagnosing a mechanism during continuous operation in which sensor readings are the primary source of information. Although the operations scenario shown in Figure 1.2 may conjure up images of a technician sitting in a control room of an industrial plant, the intended meaning here is much broader — it may include a nurse monitoring a patient in a surgical intensive care unit or a flight engineer monitoring the condition of a jet engine during a flight.

This research does *not* apply to discrete-event dynamic systems (such as digital electronic circuits) or probabilistic systems. Also, this research is not directed at "maintenance diagnosis" in which a technician may apply arbitrary input signals and probe arbitrary points in the system.

1.2 The Approach

1.2.1 Goals and Non-Goals

The general goal of this research has been to assist process operators in tasks which humans perform poorly, whether due to boredom (in the case of monitoring) or to complexity (in the case of diagnosis). However, we do not aim to replace the operator, for he/she can detect symptoms from sight, sound, and smell that are not detectable by sensors, and can make decisions based on a broader knowledge of the world than is embodied in any automated system.

The general goal of this research has been to improve the design of operator advisory systems for deterministic continuous dynamic systems. A more specific objective has been to improve the monitoring and diagnostic capabilities of operator advisor systems in a way that yields specific guarantees of performance. This research has subsequently focused on three areas: the development of a new model-centered architecture, modeling and simulation with incomplete quantitative knowledge, and the discovery of conservative discrepancy-detection methods.

Some non-goals should be understood from the outset. This research has *not* attempted to handle real-time constraints on diagnosis, generate causal explanations of misbehavior, nor recommend or perform control actions. Also, it does not address the problem of optimal sensor placement or selective sensor focus. These are all important areas of research, and some natural extensions to this work may help advance the engineering practice in these areas. This research has also not attempted to develop techniques for noise filtering, but we believe that recent work by Cheung and Stephanopoulos [CS90a, CS90b] in representing process trends with triangular episodes makes a significant contribution toward noise filtering and may be readily integrated since it is built upon the same formal approach to qualitative simulation, Kuipers' QSIM [Kui86]. Finally, this research has not addressed the important issue of automated model-building, but our approach is designed to capitalize on work in component-connection models, such as that of Franke and Dvorak [FD90].

1.2.2 Diagnosis as Model Modification

The key cognitive skill for a process operator is the formation of a mental model that not only accounts for current observations but also enables him/her to predict near-term behavior and predict the effect of possible control actions. This observation underlies our architecture for process monitoring, named MIMIC [DK89, DK91]. The basic idea is quite simple: *mimic* the physical system with a predictive model, and when the system changes behavior due to a fault or repair, change the model accordingly so that it continues to give accurate predictions of expected behavior.

Intuitively, MIMIC incrementally simulates a model of the physical system in step with incoming observations, making the state of the model track the state of the physical system. This is the paradigm of "monitoring as model corroboration". (This is similar in principle to Kalman filtering, but we will show in Chapter 2 that there are fundamental differences (and benefits) in our approach.) When observations disagree with predictions, model-based diagnosis determines the possible fault(s). When a fault is hypothesized, it is injected into the model so that the model's predictions continue to track observations. This is the paradigm of "diagnosis as model modification" or, as Simmons and Davis call it in their Generate-Test-Debug approach [SD87], "debugging almost right models".

1.2.3 Semiquantitative Simulation

Any form of model-based reasoning is fundamentally empowered (and also limited) by the type of model used. MIMIC uses a qualitative-quantitative model, hereafter referred to as a *semiquantitative* model, based on the work of Kuipers, Berleant, and Kay [Kui86, KB88, Ber91, Kay91]. Semiquantitative models provide a level of description that is intermediate between abstract qualitative models and precise numerical models. Despite having less-than-precise information, semiquantitative models are capable of impressive predictive power. For example, Widman showed that for a variety of cardiovascular disorders, a qualitative model with semiquantitative specification for just a few key parameters was able to correctly predict (qualitatively) the values of the measured variables [Wid89]. Semi-quantitative modeling and simulation provides two key benefits for monitoring and diagnosis:

1. *Reasoning with incomplete information.* Most real-world information about mechanisms is imprecise, even when we know the exact design. A semiquantitative model allows the modeler to express what is known without making inappropriate assumptions or approximations, and simulation yields ranges rather than point values. These ranges are guaranteed upper and lower bounds, enabling simple, unambiguous matching against physical measurements.

2. *Finding all behaviors automatically.* Given incomplete information about a mechanism, it is possible that the mechanism may exhibit more than one qualitatively

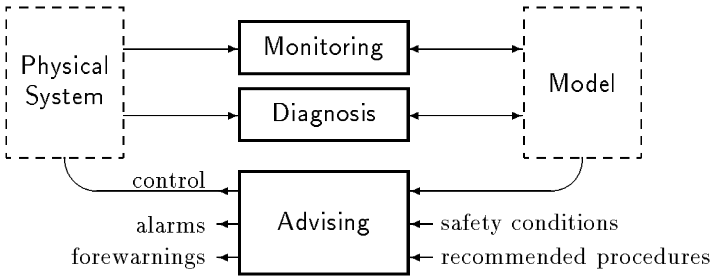


Figure 1.4: In the MIMIC architecture, three tasks mediate between the physical system and its model.

distinct behavior, such as a tank that either overflows or not. Semiquantitative simulation reveals all of the behaviors that are consistent with the incomplete information. This is especially important when trying to predict the effects of a fault or the effects of interacting faults.

1.2.4 Benefits

A key benefit of the model-based approach is that we can use the model as a window into the physical system. Specifically, the model can be used to:

- detect early deviations from expected behavior, much more quickly than with fixed-threshold alarms;
- predict the values of unobserved variables to permit alarms or other inferences on unseen variables, and to assist the operator's understanding of process conditions;
- discriminate among competing hypotheses by comparing the evolving effects of a fault against the predicted effects;
- predict ahead in time, thus forewarning of near-term undesirable or hazardous conditions;
- accumulate multiple faults over time and still give bounded behavior predictions for the degraded system; and
- predict the effect of proposed control actions to see if the control action will have the desired effect — a valuable capability in complex systems.

1.2.5 Architecture

The basic architecture of MIMIC is shown in Figure 1.4 in which a predictive model mimics the physical system. Two tasks maintain the model. The *monitoring* task advances the state of the model in step with observations from the physical system, and detects discrepancies between predictions and observations. The *diagnosis* task, upon getting a discrepancy and hypothesizing a particular fault, injects that fault into the current model to test it for consistency with current and future observations. Since a given misbehavior might be caused by one of several faults, MIMIC actually maintains a *set* of candidate models, called the *tracking set*. Each element of the tracking set represents a possible condition of the system, *i.e.*, its state and faults.

The end purpose of monitoring and diagnosis is *advice* — advice to the operator about what's happening and what to do about it. The role of the *advising* task is to apply the expert knowledge of safety conditions, recommended operating procedures, and performance objectives to produce advice in the form of alarms, forewarnings, and recommended actions. The advising task is a major beneficiary of the model-based approach in that the candidate models (and their tracked states) can provide a testbed for generating forewarnings and for testing proposed control actions.

1.3 Scope

This section characterizes the scope of the dissertation in terms of assumptions made, modeling issues, and the implementation and evaluation of the ideas.

1.3.1 Assumptions

This section describes the assumptions made in the design and operation of the monitoring and diagnostic reasoning. These assumptions include:

- The mechanism is deterministic in nature, not probabilistic.
- The dynamics of the mechanism can be modeled with ordinary differential equations, at least as an approximation. Thus, the mechanism may have state and contain feedback.
- The behavioral model of the mechanism is not invertible, in general. That is, the model can predict from inputs to outputs, but not necessarily from outputs to inputs.
- The mechanism can be described, both functionally and structurally, as a set of components and connections.
- The behavioral model of a component should define normal and fault modes which, collectively, cover the entire behavior space of the real component.

- Automatic sensor readings are the primary source of information about the state of the mechanism. There is limited opportunity to measure other variables or to perturb inputs and observe effects.
- Faults appear one-at-a-time with respect to the sampling rate for readings.
- Diagnosis must be performed while the mechanism operates.
- The mechanism may continue to operate in a degraded mode with multiple faults, so single-fault diagnosis is inadequate.
- Sensor readings may contain random noise of known magnitude.
- Incomplete knowledge of the mechanism does not imply random behavior within designated bounds. A landmark value is constant; its exact value is known only within a specified range; it does not vary randomly within that range. Likewise, functional relations are fixed, existing somewhere within the bounds specified by envelope functions; the true relation does not vary within that space.

1.3.2 Non-requirements

- The Mimic approach is not restricted to near-equilibrium behavior of a mechanism. In fact, dynamic changes in behavior are a valuable source of clues for monitoring and diagnosis.
- Observations do not have to be periodic — they can vary in frequency.
- The set of measured quantities does not have to be constant; it can change with every set of readings. This permits focusing on a subset of sensors in one phase of operation, then shifting to other subsets during other phases. Sensors known to be bad can be excluded at any time.
- Manual observations can be supplied at any time; they do not have to synchronized with automatic observations.
- Observations do not have to be supplied in temporal order. For example, in medicine, lab reports take much longer to arrive than direct monitor data applying to the same time-point.

1.3.3 Modeling Issues

Table 1.2 illustrates where this work stands with respect to several issues of modeling, simulation, fault detection, and diagnosis. Three issues require explanation:

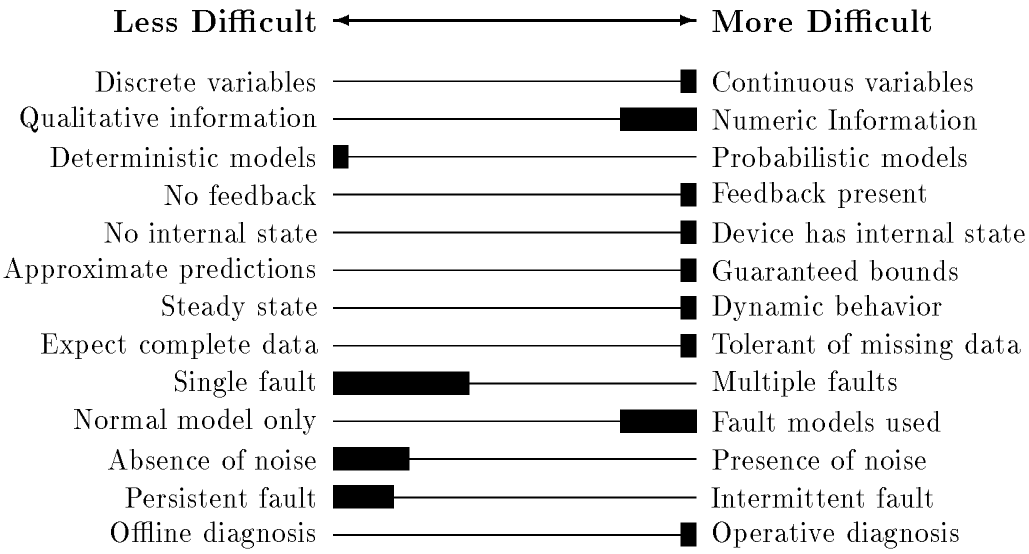


Table 1.2: Modeling issues addressed in this report. The black bars represent where MIMIC stands.

- On the issue of single-fault versus multiple-fault diagnosis, MIMIC generates only *single-change* hypotheses from a set of symptoms, but those changes are made to models that may already embody faults, so multiple-fault hypotheses are built incrementally over time, one fault at a time.

- The effects of sensor noise are ignored in the initial presentation of discrepancy-detection methods in this report. Later, we show how random noise of known magnitude weakens these discrepancy-detection methods, but does not cause false positives.

- In our method of continuous monitoring, persistent faults (when diagnosed) are corroborated over time by a stream of compatible sensor readings. Although intermittent faults cannot attain that kind of corroboration, the history of their being repeatedly hypothesized and refuted can be collected as evidence of intermittent faults.

1.3.4 Implementation

The ideas for monitoring and operative diagnosis presented in this dissertation are implemented in a computer program named MIMIC, written in Common Lisp and tested on a Symbolics 3650 computer. MIMIC builds upon four important pieces of research: Kuipers' QSIM for qualitative modeling and simulation [Kui86], Kuipers and Berleant's Q2 for semiquantitative simulation with partial quantitative knowledge [KB88], Berleant's Q3 for the technique of state insertion at measurement instants [Ber91], and Kay's NSIM for dynamic behavior envelopes [Kay91].

1.3.5 Empirical Evaluation

Claims made in this dissertation have been tested on a series of fluid flow systems of varying complexity. The fluid flow systems were simulated in both normal and faulty configurations to produce simulated sensor readings. These readings were then given as input to MIMIC to test its ability to detect and diagnose faults during continuous operation. Chapter 4 presents these results.

1.4 Claims

This report describes a method for monitoring and diagnosis of process systems based on three foundational technologies: semi-quantitative simulation, measurement interpretation, and model-based diagnosis. Compared to existing methods based on fixed-threshold alarms, fault dictionaries, decision trees, and expert systems, several advantages accrue. The claims that follow divide into three principal categories: modeling and simulation, predictive monitoring, and discrepancy detection.

1.4.1 Modeling & Simulation

Validation. It is easier to acquire and validate a model of a mechanism than to acquire and validate a set of diagnostic rules. Although the latter approach typically yields an earlier demonstration of diagnostic capability, the knowledge base is never complete, there are no guarantees of diagnostic coverage, and performance degrades with multiple faults.

Expressive Power. Semiquantitative models provide greater expressive power for states of incomplete knowledge than differential equations, and thus make it possible to build models without incorporating assumptions of linearity or specific values for incompletely known constants. The modeler can express incomplete knowledge of parameter values and monotonic functional relationships (both linear and non-linear). By specifying conservative ranges for landmark values and conservative envelope functions for monotonic relationships, semiquantitative simulation generates guaranteed bounds for all state variables. This eliminates modeling approximations and compromises as a source of false positives during diagnosis.

Soundness. Qualitative simulation generates all possible behaviors of the mechanism that are consistent with the incomplete/imprecise knowledge. This is essential for distinguishing *misbehavior* (which is due to a fault, and thus requires diagnosis) from normal behavior, especially when there is more than one possible normal behavior. This eliminates "missing predictions" as a source of false positives during diagnosis.

1.4.2 Predictive Monitoring

Operating with Faults. Large complex systems almost always operate with faults, and it is not enough just to *detect* anomalous behavior. In order to continue safe operation, it is just as important to *predict* the effects of faults in order to forewarn of possible undesirable states and to help determine appropriate control actions.

Exploiting Dynamic Behavior. Observations of dynamic behavior over time enable stronger methods of fault detection and isolation than from a single snapshot of system output. This may seem obvious, but few fault detection schemes actually base their conclusions on a *sequence* of sensor readings taken at different times while the mechanism operates. By tracking readings against model predictions, MIMIC exploits the dynamic behavior of the mechanism to corroborate or refute hypotheses.

Early Warning. By simulating ahead in time from the current state, an operator can be forewarned of nearby undesirable states that the plant might enter. Similarly, the effects of proposed control actions can be determined by simulating from the current state of every model being tracked.

Updating the State. Sensor readings supply important information that can be used to update the state of the model. By unifying conservative reading ranges with conservative prediction ranges, the semiquantitative simulation continues to generate guaranteed bounds using the latest available information.

1.4.3 Discrepancy Detection & Diagnosis

Dynamic Alarm Thresholds. Incremental simulation of the semiquantitative model in synchrony with incoming sensor readings generates, in effect, dynamically changing alarm thresholds. Comparison of observations to model predictions permits earlier fault detection than with fixed-threshold alarms and eliminates false alarms during periods of significant dynamic change, such as startup and shutdown.

Temporal and Value Uncertainty. Because a given fault may manifest in different ways under different circumstances, methods that identify faults based on specific subsets of alarms or specifically-ordered sequences of alarms are insufficient [Mal87]. Since MIMIC matches observations against a branching-time description of predicted behavior (a description that includes all valid orderings of events), and since it tests for overlap of uncertain value ranges rather than whether or not an alarm is active, it can detect all of the valid ways in which a fault manifests.

Hypothesis Discrimination. Discrimination among competing hypotheses is automatic in MIMIC. Whenever new readings arrive, every model in the tracking set is tested for discrepancies. Thus, incorrect hypotheses can be refuted either by the mechanism's natural behavior or by its response to perturbation tests. (MIMIC does not currently give advice on what inputs to perturb).

Multiple-Fault Diagnosis MIMIC supports *continuous* monitoring of a mechanism because it updates the model as faults and repairs are diagnosed. This permits incremental creation of multiple-fault diagnoses over time and continues to provide valid predictions of behavior, even though multiple faults are present.

1.4.4 Skepticism

It is reasonable, even wise, to be skeptical of new methods and the claims made for them. This section attempts to answer some of the obvious questions that might occur to the skeptical reader.

- *Why this model-based approach? Does it have any fundamental theoretical advantages over existing methods?*

  Most of the advantages are a consequence of *analytical redundancy* — the use of a process model to estimate the values of process state variables $X(t)$ and process parameters $\Theta(t)$ based on measurable inputs $U(t)$ and outputs $Y(t)$. Compared to range-checking of output variables, Isermann [Ise89] notes several advantages to this approach:

  1. In terms of signal flow, the state variables $X(t)$ or process parameters $\Theta(t)$ are, in many cases, closer to the process faults. Hence the process faults may be detected earlier and localized more precisely than by range checking of $Y(t)$.
  2. A process fault usually causes changes of several output variables $\Delta Y_i(t)$ with different signs and dynamics. The model-based fault detection now takes into account all these detailed changes, provides a data reduction and determines (theoretically) the state variable or process parameter which has been changed directly by the fault. Hence, it can be expected that a significant change $\Delta X_j(t)$ or $\Delta\Theta_j(t)$ can be extracted and the fault detection selectivity will be improved.
  3. Closed loops generally compensate for changes $\Delta Y_i(t)$ of the outputs by changing the inputs $U(t)$. Therefore deviations caused by faults cannot be recognized by range-checking alone. Model-based fault detection methods automatically consider the relations between inputs and outputs and are therefore also applicable to closed-loop systems.
  4. Model-based fault detection methods need, in principle, only a few robust sensors. This lessens the need (in traditional methods) for numerous sensors that measure,

as directly as possible, all pertinent variables. The use of numerous sensors can become expensive.

- *If this is such a good idea, why hasn't it been done before?*

  The overly simple answer is that it *has* been done before, though not in the same way as described in this report. As we will see in Chapter 2, a few model-based approaches to operative diagnosis have arisen in the last few years, and they all lend support to the basic concept. However, these efforts are too new to be in common usage. Kitamura describes current practice this way:

  > *FDI [fault detection and isolation] techniques currently in use by the industrial sector are conservative and traditional. Typical examples include calibration against standards, limit checking, mutual consistency checking (majority voting), surveillance during periodic shutdown and perturbation tests before restart.* [Kit89]

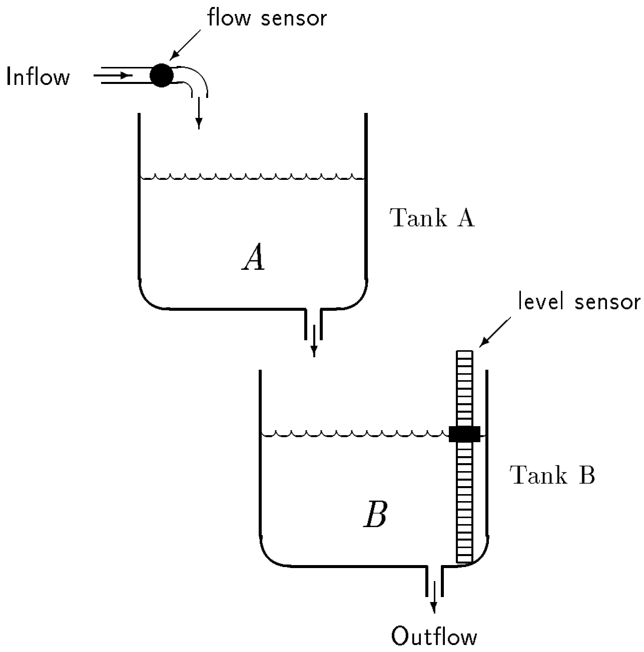
  Much of the engineering effort in the process industries has been directed not at fault detection but at automation and control, with many improvements in sensors, actuators, displays, feedforward and feedback control, and optimization. In describing the current practice in alarm monitoring and protection, Isermann notes that:

  > *...the implemented methods are still rather simple and consist mainly of limit-value checking of some easily available single signals. In contrast to the field of control, methods based on modern dynamic systems theory are hardly applied.* [Ise89, p. 254]

- *Surely there must be some drawbacks to your method, no?*

  Yes. The benefits of our method come at a price: a substantial computational load from simulating multiple models in real-time. Ten years ago this approach would have been absurdly expensive and/or unreasonably slow, given the existing computer technology. In the last ten years the price/performance ratio of computers has improved enormously and the emergence of qualitative and semiquantitative simulation technology has brought the approach into the realm of the possible. Of course, we do not claim that the approach is practical in all cases. There are certainly existing processes and mechanisms whose complexity exceeds our current capabilities.

  There are other limitations which are discussed in Chapter 5. Some of these are due to current limitations of the simulation technology; others concern issues that have always been problematic, such as reasoning with noise and failing to detect subtle faults.



Differential equations:

$$A' = inflow - f(A)$$

$$B' = f(A) - g(B)$$

Figure 1.5: Two-tank cascade. Water flows into Tank-A at a measured rate, which then drains into Tank-B, which drains to the outside. The level of water in Tank-B is measured, but the level in Tank-A is not.

1.5 Example: A Two-Tank Cascade

This section illustrates MIMIC by example, demonstrating several important properties and claims of this work. The intent here is to show *what* MIMIC does, not necessarily *how* it does it. The ideas presented here will be developed more fully in Chapter 3. We begin by describing a simple mechanism and show how it is modeled in MIMIC. Then, we examine what happens as a fault occurs, as MIMIC monitors and diagnoses the mechanism.

1.5.1 Modeling

Consider the two-tank cascade shown in Figure 1.5. Water flows into the top of Tank-A at a measured rate; Tank-A drains into Tank-B, whose level is measured; and Tank-B drains to the outside. If everything is working normally and a constant inflow is applied, the level in each tank will reach equilibrium (assuming that no overflow occurs).

Note that there is no sensor for the level in Tank-A; this state value is unmeasured, so an operator has no way of knowing its level except through manual measurement. Likewise, there is no sensor on the outflow from Tank-B, so an operator cannot directly determine if outflow is normal.

The scenario begins with both tanks empty. A constant inflow is applied and the

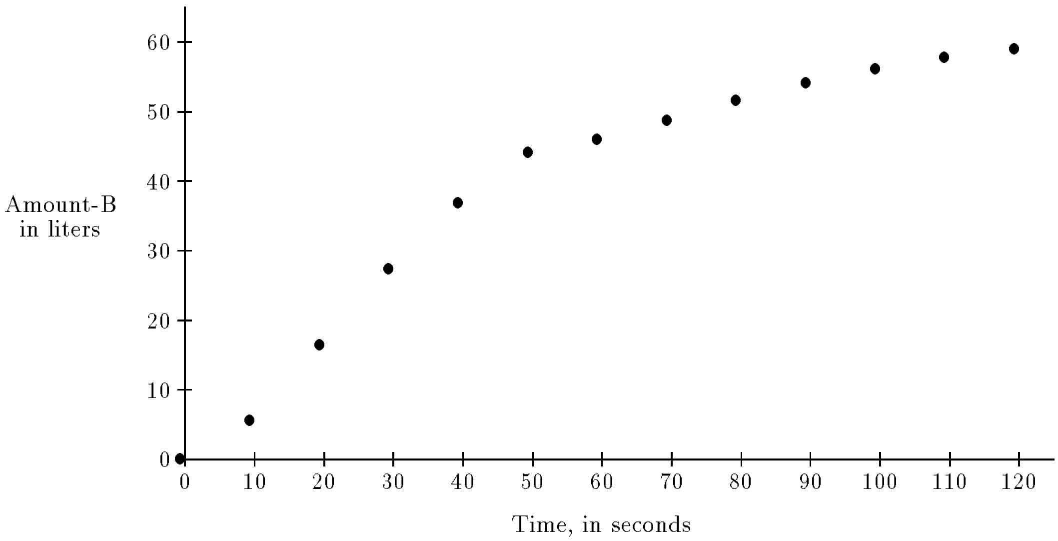


Figure 1.6: Raw sensor readings from the two-tank cascade. The readings are affected slightly by a partial obstruction in the drain of Tank-A at $t = 50.01$.

tanks begin filling toward equilibrium. Everything works normally until $t = 50.01$, at which time the drain of tank-A becomes partially clogged, reducing its flow rate by 20%. Note that this fault does not alter the basic dynamics of the system; it still functions as a 2-tank cascade and the slight change in behavior is barely noticeable in the raw sensor readings shown in Figure 1.6.

The challenge, of course, is to detect and diagnose the fault. Using the paradigm of "monitoring as model corroboration", a semiquantitative model of the mechanism predicts the mechanism's behavior, which is then compared to observations. Figure 1.7 shows the model for the two-tank cascade, from which all predictions are made. Incomplete knowledge of the physical two-tank cascade is expressed in this model as numeric ranges for landmark values (see the `initial-ranges` clause). Incomplete knowledge can also take the form of envelope functions for monotonic relations, though none are required in this example.

1.5.2 Simulation

Figure 1.8 shows the actual behavior of our faulty two-tank cascade overlaid on the predicted behavior of a normal two-tank cascade. The short vertical lines show the range of each reading. Sensors are not exact measuring devices, but are designed to yield a measurement correct to within a specified tolerance, such as $\pm 3\%$. Thus, the sensor reading ranges result from applying the tolerance to the actual reading value. The rectangles show the predicted range of a variable (there is no temporal range in the prediction; the rectangle has width only for display purposes). Here, the range results from the the partial

```
(define-QDE TWO-TANK-CASCADE
  (quantity-spaces
    (Inflow-A    (0 normal inf)    "flow(out->A)")
    (Amount-A    (0 full)          "amount(A)")
    (Outflow-A   (0 max)           "flow(A->B)")
    (Netflow-A   (minf 0 inf)      "d amount(A)")
    (Amount-B    (0 full)          "amount(B)")
    (Outflow-B   (0 max)           "flow(B->out)")
    (Netflow-B   (minf 0 inf)      "d amount(B)" )
    (Drain-A     (0 vlo lo normal))
    (Drain-B     (0 vlo lo normal)))

  (constraints
    ((MULT Amount-A Drain-A Outflow-A)   (full normal max))
    ((ADD  Outflow-A Netflow-A Inflow-A))
    ((D/DT Amount-A Netflow-A))
    ((MULT Amount-B Drain-B Outflow-B)   (full normal max))
    ((ADD  Outflow-B Netflow-B Outflow-A)) ; Outflow-A = Inflow-B
    ((D/DT Amount-B Netflow-B)))

  (independent   Inflow-A Drain-A Drain-B)
  (history       Amount-A Amount-B)
  (unreachable-values
    (netflow-a minf inf) (netflow-b minf inf) (inflow-a inf))
  (initial-ranges
    ((Inflow-A    normal) (3.01   3.19))        ; +/- 3% of 3.1
    ((Amount-A    full)   (99     101))
    ((Amount-B    full)   (99     101))
    ((Time   T0)          (0      0))
    ((Drain-A normal)     (0.0485 0.0515))      ; +/- 3% of .05
    ((Drain-B normal)     (0.0485 0.0515))      ; +/- 3% of .05
    ((Drain-A lo)         (0.020  0.0485))
    ((Drain-B lo)         (0.020  0.0485))
    ((Drain-A vlo)        (0      0.020))
    ((Drain-B vlo)        (0      0.020))))
```

Figure 1.7: Semiquantitative model of the two-tank cascade. Incomplete quantitative knowledge is expressed in the form of ranges for landmark values, in `initial-ranges`. When models contain monotonic function constraints (this one does not), incomplete knowledge can also be expressed in the form of envelope functions.

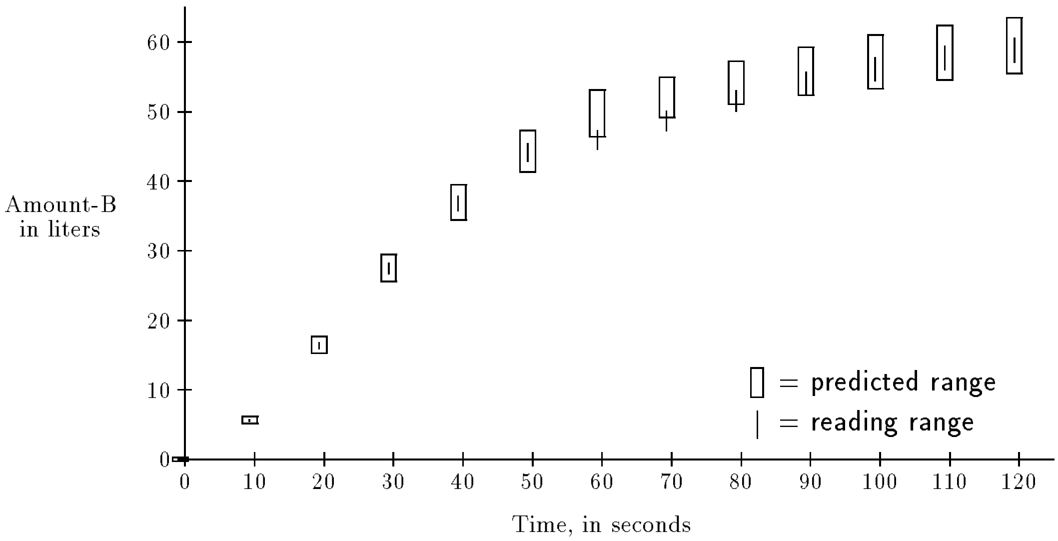


Figure 1.8: Sensor readings and fault-free predictions from the two-tank cascade. The simple limit test fails to detect any discrepancy between observations and predictions.

quantitative knowledge in the model; the semiquantitative simulation methods generate upper and lower bounds for each variable.

The small "kink" in the predicted range at $t = 70$ is a consequence a unifying readings with predictions on each cycle. Specifically, the small intersection at $t = 60$ properly altered the next prediction for $t = 70$, narrowing its range and lowering its value.

1.5.3 Discrepancy Detection

This dissertation will describe four methods for detecting a fault by detecting incompatibilities between predictions and observations. The first and most obvious method is to test for overlap between a reading range and its predicted range. As Figure 1.8 shows for this example, there is indeed overlap for every reading, so no fault is detected with this method. Note that at $t = 60$ (the first reading after the fault) the overlap is very small, but quickly returns to "normal" within about 30 seconds. This example illustrates two important problems: for systems having compensatory response, some faults can *only* be detected in their earliest moments, and simple range-checking is not always sensitive enough to detect abnormal perturbations in behavior.

MIMIC uses three other methods of discrepancy detection, to be described later in Chapter 3. In this particular case, the fault is detected as an *analytic* discrepancy, meaning that the assumptions, predictions, and observations are *mutually* incompatible, even though they are *individually* compatible. In this case, the analytic discrepancy was discovered when the range of overlap for `Amount-B` at $t = 60$ was asserted back to the model

variable	*mode value*	*fault type*	*probability*
`Drain-A`	`normal`	`nil`	.95
	`lo`	`abrupt`	.04
	`vlo`	`abrupt`	.01
`Drain-B`	`normal`	`nil`	.95
	`lo`	`abrupt`	.04
	`vlo`	`abrupt`	.01

Table 1.3: Some alternate operating modes in the two-tank cascade.

and its effects propagated through the model; it was inconsistent for the upper bound of `Amount-B` to be so low (47.30) given the predicted range for `Amount-A` ([55.79 62.19]) and the assumed ranges for `Drain-A` ([.485 .515]) and `Drain-B` ([.485 .515]). This demonstrates an important strength of the model-based approach in general and the semiquantitative methods in particular: by using constraint-like descriptions, the model can be used not only to predict behavior from an initial state, but also to check the mutual consistency of a set of readings that otherwise appear to agree with the model's predictions.

1.5.4 Hypothesis Generation

Up until this time the tracking set has contained only the fault-free model. The discrepancy with `Amount-B` at $t = 60$ removes that model from the tracking set and initiates hypothesis generation via dependency tracing through a structural model of the mechanism. The structural model, shown in Figure 1.9, is essentially a declarative description of the mechanism's components and connections, shown graphically in Figure 1.5. By tracing upstream from the site of the discrepancy (`Amount-B`), MIMIC identifies the components and parameters whose malfunction could have caused the discrepancy. In this case, the suspects are `Amount-B-sensor`, `Tank-B`, `Drain-B`, `Tank-A`, `Drain-A`, and `Inflow-A`. To keep this example simple, we'll focus on two suspects: `Drain-A` and `Drain-B`.

Having identified the suspects, MIMIC consults a table of alternate operating modes for each suspect in order to generate modifications of the current model. Table 1.3 shows the possible mode values for `Drain-A` and `Drain-B`. These values are present in the semiquantitative model (figure 1.7) and can therefore be used to instantiate a modified version of the current model. The fault type can be either `abrupt` or `drift`. All abrupt faults are always hypothesized since these faults can occur at any time. For drift faults, however, only the one or two drift fault values adjacent to the current value are hypothesized (because the fault represents drift from the normal value). The *a priori* probability of each mode is used in ranking hypotheses (and as we will see in Chapter 3, hypotheses can also be ranked by age and by degree-of-match).

Component definitions:

```
(COMPONENTS
   (INFLOW-SENSOR
      (flow-in   input  icon1)
      (measured  output i-obs))
   (TANK-A
      (inlet     input  icon1)
      (outlet    2-way  tocon1)
      (drainrate input  dr1))
   (TANK-B
      (inlet     input  tocon1)
      (outlet    2-way  tocon2)
      (drainrate input  dr2)
      (amount    output acon))
   (AMOUNT-SENSOR
      (amount-in input  acon)
      (measured  output a-obs)))
```

Parameter to connection mappings.

```
(PARAMETERS
   (Inflow-A   icon1)
   (Drain-A    dr1)
   (Drain-B    dr2))
```

Variable to connection mappings.

```
(VARIABLES
   (Amount-B   a-obs)
   (Inflow-obs i-obs))
```

Figure 1.9: Structural model of the two-tank cascade. This model is used by the dependency tracer to trace upstream from the site of a discrepancy to identify all the components and parameters whose malfunction could have caused the discrepancy.

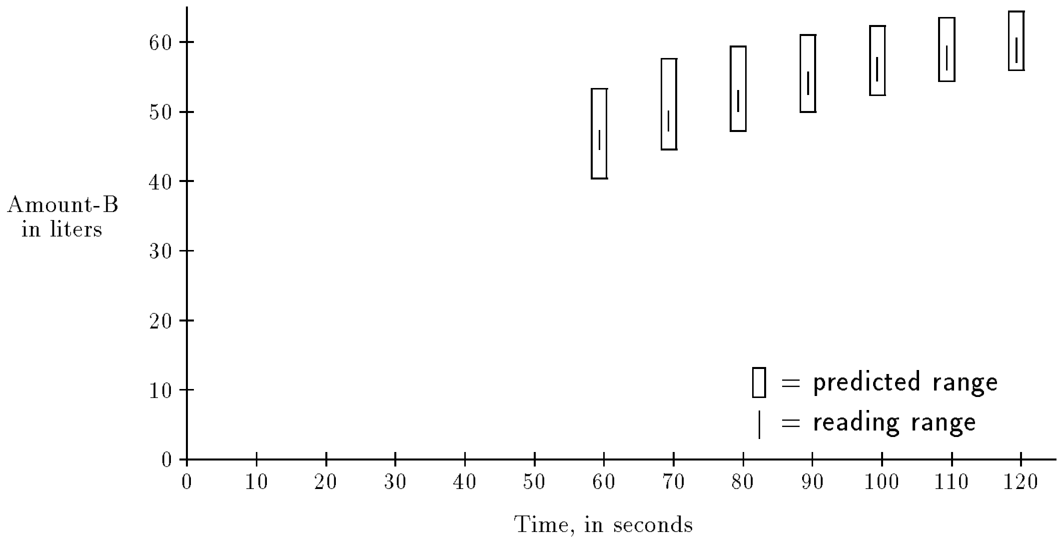


Figure 1.10: Predictions for Amount-B showing agreement with the hypothesis that Drain-A is partially clogged .

1.5.5 Hypothesis Testing

MIMIC assumes that faults occur one-at-a-time with respect to the sampling rate for readings, so it hypothesizes single-changes to the existing model. One model is created for `Drain-A = lo`, another for `Drain-B = lo`. For each new model, MIMIC attempts to initialize it using the hypothesized modes, current readings, and latest predictions. Since there may have been some time elapsed between the moment that the fault occurred and the moment that it manifested as a symptom, MIMIC performs a procedure termed *resimulation.* This procedure reattempts initialization at successively earlier reading moments, simulating the model up to the current time and quantifying the similarity between predictions and readings. The initialization time that yields the strongest similarity during this hill-climbing search is taken as the probable time of failure. Resimulation is essential because it revises the values of unobserved state variables to reflect the effects of an earlier fault. Without resimulation, future predictions of behavior would be incorrect.

In the case of `Drain-B = lo`, the model fails to initialize. Specifically, every attempt to initialize it results in an inconsistency, meaning that the fault hypothesis embodied in this model is immediately falsifiable. In general, though, some incorrect fault models will initialize and be carried in the tracking set until future readings show it to be incompatible. In the case of `Drain-A = lo`, the model initializes successfully and resimulation yields the time-of-failure to be $t = 50$. Figure 1.10 shows the predicted ranges for this model. The wider predicted ranges for this model versus the normal model are a consequence of the wider range for `lo` ([.02 .0485]) versus the narrow range for `normal` ([.0485 .0515]).

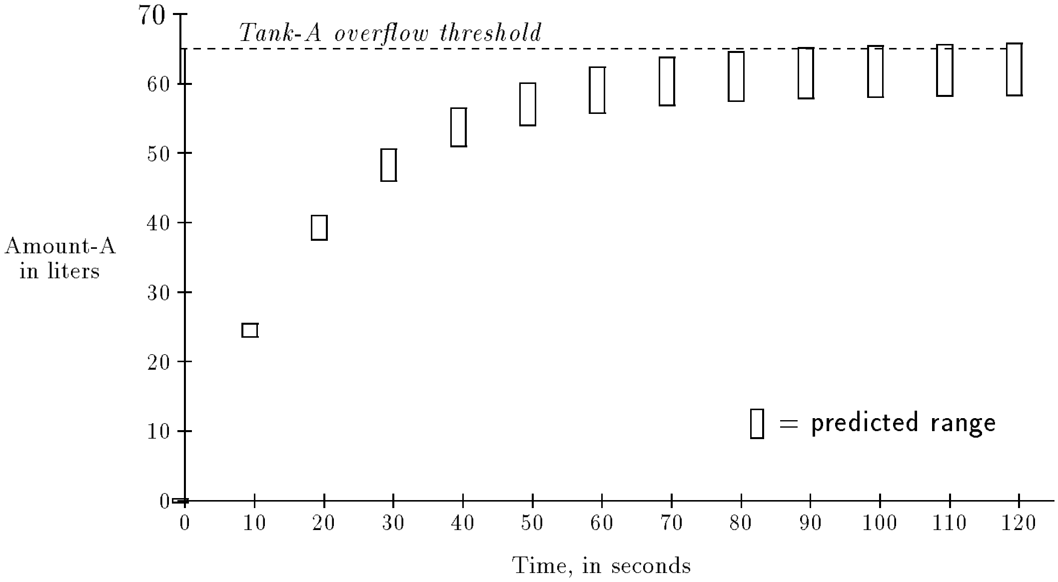


Figure 1.11: Overflow prediction for Tank-A. The level in Tank-A is not observable and can only be inferred using the model. Given the hypothesis that Drain-A is partially clogged, prediction shows that overflow can occur as early as $t = 91.6$

1.5.6 Forewarning

For monitoring, an important advantage of the model-based approach is that it predicts ranges for unobserved state variables. In our two-tank cascade, there is no level sensor for Tank-A, so there is no way for the operator to know the amount in Tank-A. However, by simulating in parallel with the mechanism a model that embodies all diagnosed faults, the operator can be kept informed of the values of unobserved variables. Furthermore, by simulating the model ahead in time, the operator can be forewarned of future undesirable states. Figure 1.11 shows the predicted ranges for the amount in Tank-A. Let's assume that the capacity of Tank-A is 65 liters, as shown with the dashed line. After the diagnosis of `Drain-A = lo` at $t = 60$ (and the corresponding change in the model), predictions show that Tank-A could overflow as early as $t = 91.6$. In general, the tracking set may contain more than one model, so forewarnings are based on the earliest warnings from the set of models.

1.5.7 Summary

Let's summarize what has happened. MIMIC began tracking the fault-free model at $t = 0$, and as each new set of readings appeared, it tested for discrepancies between predictions and readings using four distinct methods. It also unified predictions with observations, yielding tighter predictions of future behavior. At $t = 50.01$ the drain of Tank-A became partially obstructed, and the fault was detected at the next readings at $t = 60$ as an analytic discrepancy. Two fault hypotheses were formed and instantiated as modifications of the current fault-free model. One hypothesis was immediately discarded because it failed to initialize, but the other initialized successfully and was resimulated from successively earlier moments, localizing the time-of-failure to be $t = 50$. This adjusted the unobserved state values (`Amount-A`, in this case) to reflect the effects of the fault between $t = 50$ and $t = 60$. Subsequent predictions from the fault model were corroborated by the readings and also used to forewarn the operator of a possible Tank-A overflow at $t = 91.6$.

1.6 Guide to the Dissertation

This dissertation is organized into five chapters. This chapter has introduced the problem of monitoring dynamic systems and briefly demonstrated a new approach and its benefits. Chapter 2 describes existing methods for monitoring and diagnosis, showing their similarities and differences with the MIMIC approach. Chapter 3 presents our contributions to the theory of fault detection in dynamic systems and to the engineering design of automated process monitoring. Chapter 4 presents results from an implementation of this theory, describing the diagnostic program MIMIC and its performance on a set of fluid-flow problems. Chapter 5 discusses the implications of this work and presents ideas for future research.

1.7 Terminology

This section defines some terms used throughout this report.

abrupt fault An abrupt fault has a sudden effect on the mechanism, such as the sudden failure of a pump. Sometimes called *cataleptic* fault. Contrast with *incipient fault.*

accommodation As part of the task of *monitoring*, accommodation of a faulty mechanism to normal operation requires correcting or compensating for the effects of a fault.

analytical redundancy The term *analytical redundancy*, also called *functional redundancy*, refers to the fault detection method of using known analytical relationships among sets of signals, such as outputs from dissimilar sensors, to check for mutual consistency. The method (and the phrase) emerged as an alternative to the earlier practice of *hardware redundancy*, wherein 3 or 4 identical sensors and voting logic are used for fault tolerance.

attainable envisionment An *attainable envisionment* of a mechanism is the set of all qualitatively distinct behaviors possible from a given initial state. As generated by QSIM, an attainable envisionment is a tree whose root node is the initial state and where each directed link represents a transition to a valid successor state node. When a state has more than one valid successor state, the resulting branch indicates a qualitative distinction in behavior. Any single path through the tree is a behavior, represented as a sequence of states. Contrast with *total envisionment.*

behavior The observed behavior of mechanism is its sequence of sensor readings. The predicted behavior of a mechanism is a sequence of states in which each state contains values for all state variables and is justified as a valid successor of the preceding state. See *attainable envisionment.*

candidate A candidate is a model of the mechanism whose predictions are compatible with the current readings. A candidate, therefore, embodies a hypothesis about the operating mode of each component, whether normal or faulty.

candidate generation When discrepancies are found between the observed behavior and the behavior predicted by the mechanism model, candidate generation produces one or more possible explanations for those discrepancies in the form of modifications of the mechanism model.

component A component is any piece of a mechanism, such as the level sensor in the two-tank cascade. The concept of a component is hierarchical; an entire steam turbine may be regarded as a component of a larger mechanism, just as the fan blades are components of the steam turbine.

consequential fault A consequential fault, sometimes called an induced fault, is a defect caused by an earlier fault. For example, if a pump motor fails in a way that causes it to draw an excessive amount of current, that may cause a fuse to blow as a consequence of the excessive current. Some consequential faults propagate rapidly, creating the appearance of multiple simultaneous failures.

defect A defect is a fault whose cause is internal to the mechanism, such as a component that is broken or out of calibration, or a connection that is severed or blocked. Diagnosis of a defect calls for repair of the mechanism. Contrast with *disturbance.*

diagnosis A diagnosis is a plausible explanation for an *unexpected* behavior. Note that behavior which is undesirable but predictable (such as subjecting a fault-free mechanism to a known overload) does not need to be diagnosed.

discrepancy A discrepancy is an incompatibility between an observation (whether direct or derived) and a prediction. For example, if the predicted range for x is [2.1 2.4] and the reading range is [2.5 2.6], then there is a discrepancy. A discrepancy that cannot be resolved (such as by advancing the simulation) becomes a *symptom.*

discrepancy detection Sometimes called fault detection, discrepancy detection is the task of detecting misbehavior in the mechanism, as in recognizing when observed behavior differs from expected behavior. It does not include identifying the cause (see *fault isolation*).

discrimination As diagnosis proceeds there are usually several candidates that could explain all the discrepancies. Discrimination is the process of gathering additional observations from the mechanism in order to refute incorrect candidates through discrepancy detection.

disturbance A disturbance is a fault whose cause is external to the mechanism, such as an abnormally high ambient temperature. Diagnosis of a disturbance calls for changes in environmental conditions rather than repairs to the mechanism. Contrast with *defect.*

drift fault Same as *incipient fault.*

false negative In fault detection, a false negative is the failure to detect a fault when one is present. There are several fundamental reasons why a fault may be undetectable: the fault may be masked by a redundant spare; the fault may not be exposed in the current operating mode, such as a burned-out light bulb with no power applied; the fault may not have affected any sensor (yet), or the affected sensor may itself be defective; the fault manifestations may be buried in noise or may simply be too small to distinguish from normal behavior.

false positive In fault detection, a false positive (sometimes called a false alarm) is the "detection" of a fault when none is present. This can occur in at least three ways: when readings from the fault-free mechanism exceed a detection threshold, whether due to noise or to acceptable variations within the mechanism; when thresholds designed for steady-state operation are triggered during other phases of operation, such as startup and shutdown; and when a normal-but-infrequent behavior is neglected in threshold design, such as the opening of a pressure-relief valve.

fault A fault is an abnormality in a mechanism or its environment. A fault is either a *defect* in a component or a *disturbance* in an exogenous variable or parameter. The onset of a fault may be abrupt *(abrupt fault)* or gradual *(incipient fault).* A fault does not necessarily manifest in a symptom; the fault may not be exposed in certain operating modes (such as a burned-out light bulb with no power applied) or its effects may be masked by a redundant spare.

fault detection Same as *discrepancy detection.*

fault isolation Fault isolation is the task of localizing a fault to a specific component or external input of the mechanism.

fidelity A model has fidelity when it does not support incorrect predictions about the mechanism. Compare with *precision*.

functional redundancy Same as analytical redundancy.

incipient fault A small or slowly developing fault is often called an *incipient* or *evolving* or *drift* fault. Such faults may be due to aging or drift. Contrast with *abrupt fault*.

Kalman filter The Kalman filter can be thought of as a processor that produces three types of output, given a noisy measurement sequence and associated models. First, it can be thought of as a *state estimator* or *reconstructor*, i.e., it reconstructs estimates of the state $x(t)$ from noisy measurements $y(t)$. Second, the Kalman estimator can be thought of as a *measurement filter* which accepts the noisy sequence $\{y(t)\}$ as input and produces a filtered measurement sequence $\{\hat{y}(t|t)\}$. Third, the filter can be thought of as a *whitening filter* that accepts noisy correlated measurements $\{y(t)\}$ and produces uncorrelated or white-equivalent measurements $\{e(t)\}$, the innovation sequence. [Can88, p. 321]

mechanism A mechanism is a physical system which has structure and whose behavior and state is the object of operational attention. Specific application domains may refer to the mechanism as a "device" or "process" or "system" or "patient".

mode A component may operate in one of several modes, such as a thermostat that is either *on* or *off*. A mode variable may change value automatically as specified in the model (such as when a thermostat changes from *off* to *on*) or as a result of a fault hypothesis (such as a thermostat that is "stuck on").

model In this report the noun *model* refers to an abstraction of the mechanism, usually as a model of structure (components and connections) or behavior (semiquantitative constraint equations). (This is different from its meaning in logic in which an interpretation I is said to be a model of a sentence ϕ if I satisfies ϕ for all variable assignments.)

monitoring Monitoring is the continuous real-time process of detecting anomalous behavior, tracking the effects of a fault, and determining control actions to continue safe operation in the presence of faults. Monitoring includes *discrepancy detection*, *fault isolation*, and *accommodation*.

operator An operator is a human responsible for the safe and efficient operation of a mechanism. An operator's duties include monitoring the mechanism's behavior, diagnosing the possible cause(s) of misbehavior, and taking corrective action to control the mechanism.

parameter A parameter is a fixed quantity in a mechanism or, analogously, a constant in a model of the mechanism. For example, the electrical resistance of a heating element is a parameter. Some defects are termed "parameter faults" because a parameter value has changed to an abnormal value.

precision A model has precision to the extent that the predictions it makes are strong enough to be falsifiable by observations of the actual mechanism. Compare with *fidelity.*

repair A repair is a change from faulty to normal. When talking about the mechanism, a repair means the disappearance of a fault, whether due to a fix or to spontaneous remission. When talking about a hypothesis, a repair is a change from a fault model of some component to its normal model.

structure The structure of a mechanism is its components and connections, usually represented as a directed graph in which the vertices are components and the edges are connections.

suspect A suspect is a component or parameter of the mechanism whose malfunction could account for a discrepancy. If the suspect is an input parameter of the mechanism, then it identifies a possible *disturbance*; if it is a component, then it identifies a possible *defect.*

symptom A symptom is an incompatibility with expected behavior, as detected in an unresolved discrepancy. A symptom is a manifestation of one or more changes (faults or repairs) in the assumed condition of the mechanism. There may be an arbitrarily long time between the occurrence of a fault/repair and its manifestation as a symptom. Whether or not a symptom is detectable depends on the size of the perturbation, the precision of the model, the placement and precision of sensors, the magnitude of noise, and the sensitivity of the discrepancy-detection methods.

total envisionment A *total envisionment* is a representation of *all* behaviors inherent in some mechanism in some configuration, for each possible initial state. Represented as a graph, each node is a possible state of the mechanism and each directed link represents a valid transition from one state to another. In design analysis, for example, a total envisionment can reveal whether there is any initial state that does not lead to the desired behavior. Contrast with *attainable envisionment.*

tracking set The *tracking set* is the set of models (and their states) that are being tracked at any given time. Candidate generation adds models to the tracking set; discrepancy detection removes models.

Chapter 2

Related Work

There is a vast amount of work published on the subjects of monitoring and diagnosis — too much to cover adequately in the space of this chapter. Our objective here is to focus on the most relevant work, which divides into three categories:

- symptom-based approaches that have been applied in the past but have been rejected in this research because of inherent limitations;
- model-based approaches that deserve close comparison because of similarities to MIMIC; and
- other research results that have been influential in the design of MIMIC.

2.1 Symptom-Based Approaches

Much of the literature on diagnosis describes methods of associational inference that relate symptoms directly to faults. This encompasses representations based on rules, decision trees, and fault dictionaries. Each of these methods has proven its worth in various diagnostic settings, but in the case of operative diagnosis, several limitations are shared to varying degrees:

- failure to exploit clues from the time-varying behavior of the mechanism;
- limited ability to diagnose multiple faults;
- no predictive power to reveal possible future effects of a fault or effects of compensating control actions.

The following sections examine each of the three methods in more detail, describing specific limitations for operative diagnosis.

2.1.1 Rule-Based Systems

Traditional rule-based systems have been built by accumulating the experience of expert troubleshooters in the form of empirical associations — rules that associate symptoms with underlying faults [DH88]. The problem-solving strategy may be either data-driven (forward-chaining) or goal-driven (backward-chaining) or a combination of both. The data-driven approach is most appropriate for the task of monitoring, where it is important to respond quickly to new readings and alarms, combining multiple pieces of evidence to assess the likelihood and severity of a problem, providing the operator with an interpretation of a possibly overwhelming amount of data. The goal-driven approach is most often used in diagnosis where the goal is a diagnostic conclusion and the rules, through backward chaining, seek supporting evidence for various intermediate and final conclusions.

The rule-based approach became popular, in part, because it permitted easy construction of expert systems by encoding heuristic knowledge in the form of if-then and when-then rules. The technology permitted rather quick and impressive demonstrations of diagnostic capability and promised that diagnostic coverage could be increased just by adding more rules. However, several limitations became apparent as rule-based systems were applied to increasingly large and complex mechanisms:

- The encoded knowledge is based on experience with the mechanism, and it may take a long time to accumulate the necessary experience before diagnostic patterns emerge. This is a particular problem for newly designed mechanisms.
- The task of knowledge engineering (i.e., extracting experiential knowledge from experts and representing it in an appropriate form) is widely acknowledged to be the bottleneck in the building of new expert systems. As the mechanism under study becomes larger and more complex, so too does the task of knowledge engineering.
- There is no guarantee that novel faults (i.e., faults not specifically considered during knowledge engineering) will be detected, much less diagnosed. Likewise, two faults that are individually diagnosable may interact in ways that mask any or all of the symptoms. A rule-based system may be validated on a set of test cases but still have important gaps in its knowledge base.
- Rule-based systems have little, if any, predictive power. They cannot show what will happen if a fault is left unrepaired or what will happen if some control action is taken to compensate. Similarly, it is difficult to express and reason about temporal information, such as the evolution of symptoms of a fault.
- Failed sensors is a problem for rule-based systems. If a rule depends on evidence from N sensors, it requires $2^N - 1$ rules to handle all combinations of failed sensors (assuming that none of the sensors are redundant).
- Different phases of operation (such as startup, normal operation, and shutdown) typically require separate sets of rules because of large differences in system behavior.
- Small changes in the design of the mechanism may necessitate revisions in a large part of the rule-base.

2.1.2 Fault Dictionaries

A fault dictionary is a list of symptom/fault pairs, indexed by symptom. The dictionary is built by simulating a model of the mechanism for every kind and combination of faults anticipated. Each simulation generates a description of how the entire mechanism would behave if a specific component were broken in a specific way. The result is a list of fault/symptom pairs, which is then inverted to form a dictionary of symptom/fault pairs.

To an extent, fault dictionaries overcome some of the limitations associated with rule-based systems: the dictionary does not depend on experience (so it can be made available quickly), it can expose the effects of interacting faults, it can be regenerated mechanically if the design changes, and it is likely to cover more fault scenarios because of its systematic (though not exhaustive) treatment of the kinds and combinations of faults.

The idea behind fault dictionaries — systematic generation of symptom/fault associations — is good, but the technique has some practical limitations:

- To make the simulation task tractable, only the most likely failure modes of each component are considered, and simulation of combination faults is severely limited. Thus, the approach usually cannot guarantee detection of all single-faults, let alone multiple faults.
- In the case of continuous-variable dynamic systems, simulations must be performed with nominal values, yielding symptoms that are only an approximation of what might be observed. Thus, there is an "approximate-matching problem" in deciding if a mechanism actually exhibits a given set of symptoms.
- In a single entry of the dictionary the symptoms are essentially a predicate on a snapshot of the observable variables of the mechanism. This kind of matching fails to exploit temporal continuity in the evolving manifestations of a fault (which can be vital in getting the right hypothesis and refuting incorrect hypotheses).

2.1.3 Decision Trees

Decision trees provide a guide to diagnosis in that they write down the sequence of tests leading to a diagnostic conclusion. Decision trees used in process industries are typically built manually by engineers using detailed knowledge of the plant's design and its known failure modes. Such decision trees may contain not only diagnostic steps but also recommended control actions to ensure plant safety, even before a diagnostic conclusion is reached.

As Davis and Hamscher [DH88] point out, the simplicity and efficiency that is a strength of decision trees is also an important weakness: they are a way of writing down a diagnostic strategy, but offer no indication of the knowledge used to justify that strategy. Decision trees thus lack "transparency" and are therefore difficult to update (a small change to the mechanism may require a major restructuring of the tree). Like the other methods, decision trees have no predictive power to reveal the propagating effects of a fault.

2.2 Model-Based Approaches

Model-based diagnostic reasoning can be viewed as an interaction between prediction and observation [DH88]. By using a behavioral model of the mechanism and viewing misbehavior as anything other than what the model predicts, model-based diagnosis covers a broader collection of faults than symptom-based approaches. By matching against predictions rather than symptomatic patterns, model-based diagnosis also avoids combinatoric problems in handling failed sensors and data that is for any reason unavailable [Sca89]. Another virtue of the technique is its device-independence, enabling reasoning about a system as soon as a structural model and behavioral model is available. Also, constraint-like descriptions of the mechanism allow both simulating its behavior and making inferences about the values of unmeasured variables.

The principles of model-based diagnosis are well understood; the diversity of work in this field owes largely to the many types of models that can be used within this framework, representing many different degrees of abstraction. Our coverage of this field therefore attempts to be representative rather than exhaustive. In the first three sections, MIMIC is compared to three other monitoring systems: PREMON, DRAPHYS, and MIDAS. Interestingly, these four research efforts arose at about the same time, apparently independently. The similarities are encouraging in that they lend support to a common set of concepts, and the differences are enlightening in terms of the distinct types and capabilities of models used.

2.2.1 PREMON/SELMON (Doyle et al.)

The concept of *predictive monitoring* is the inspiration behind PREMON, a focused context-sensitive approach to monitoring [DSA87, DSA89] and its successor SELMON [DF91]. Both MIMIC and PREMON take the position that a predictive simulation model is required for effective monitoring of a dynamic physical system. Both collect measurements from sensors and interpret that data with respect to predictions from the model. Both use models that are fundamentally qualitative but may be augmented with partial quantitative information.

Because of different objectives, MIMIC and PREMON turn out to be largely complementary. PREMON focuses on two issues: how to adjust alarm thresholds to reflect the changing operating context of the system, and how to utilize sensors selectively so that nominal operation can be verified reliably without processing a prohibitive amount of sensor data. Both issues depend on predicting the expected behavior of the system in a dynamic operating context, and require only the single fault-free model of the physical system.

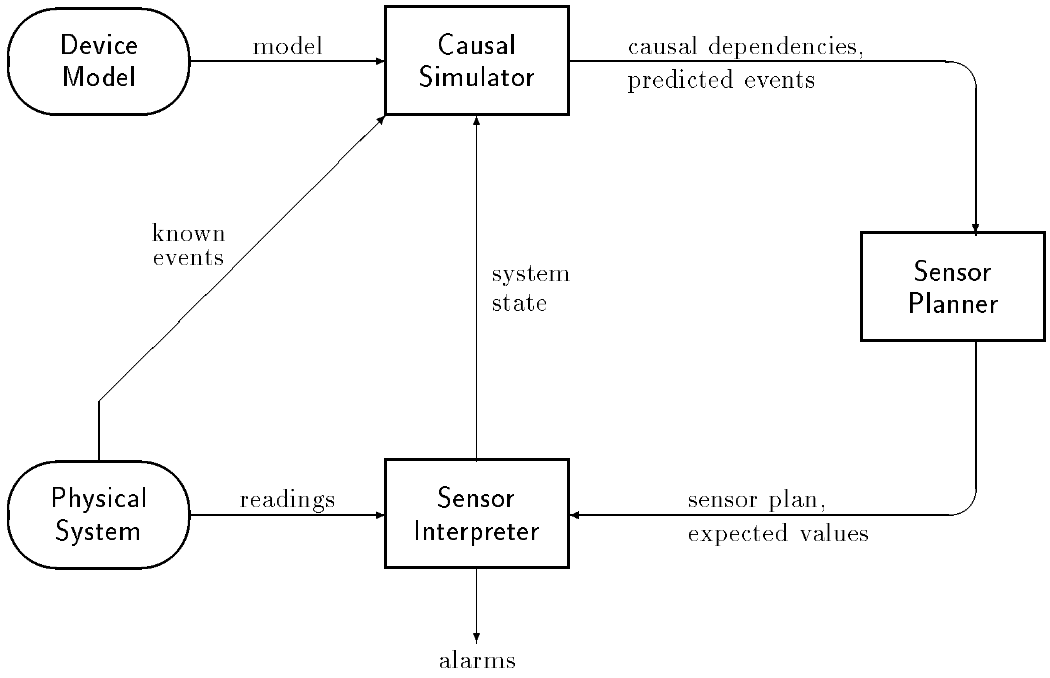


Figure 2.1: Architecture of PREMON.

PREMON uses a *predict-plan-sense* cycle using a causal simulator, a sensor planner, and a sensor interpreter, as shown in Figure 2.1. The causal simulator takes as input a causal model of the system to be monitored, and a set of events describing the initial state of the system and possibly some future scheduled events. The causal simulator produces as output a set of predicted events and a graph of causal dependencies among those events. The sensor planner takes as input that causal dependency graph and determines which subset of the predicted events should be verified. Those events are then passed on to the sensor interpreter. The interpreter compares expected values as predicted by the causal simulator with actual sensor readings. Alarms are raised when discrepancies occur. Finally, the most recent sensor readings are passed back to the causal simulator to contribute to another predict-plan-sense cycle.

In contrast, MIMIC focuses on two different issues: how to extract the most information possible from observations and predictions in order to detect discrepancies, and how to continue monitoring and continue safe operation in the presence of faults. Where PREMON uses a single fault-free model, MIMIC necessarily uses fault models in order to predict the effects of faults. MIMIC uses all available sensor data; it does not focus on a subset as PREMON and SELMON do.

2.2.2 DRAPHYS (Abbott)

Of all the related work in this chapter, Abbott's work on operative diagnosis [Abb90] is the most similar in objective to MIMIC, but differs substantially in several aspects.

DRAPHYS uses three types of models: a numerical simulation model (as a precise behavioral model), a component hierarchy (as a model of mechanism structure), and directed graph of the paths of propagation (as an abstract, qualitative model of behavior). The numerical simulation model of the fault-free mechanism is used to generate predictions of behavior. When a prediction differs significantly from observations, DRAPHYS generates a qualitative symptom containing: the qualitative value of the sensor reading as positive (+), zero (0), or negative (−); the status of the reading compared to its prediction, as high or low; the qualitative value of the derivative, and the status of the derivative compared to its predicted value. Abbott acknowledges that it is difficult to determine when a signal differs "significantly" from its expected value. Two major reasons for this are sensor noise and lack of model fidelity. The approach in DRAPHYS errs in favor of false positives.

The next step is to localize the symptoms to a subsystem of the mechanism, which is done by inspecting the component hierarchy. For example, if all of the symptoms (e.g., `Pressure-N2` and `Fuel-flow`) come from components within a common subsystem (such as `Engine-B`), then DRAPHYS localizes the problem to the subsystem `Engine-B`. Each component in the engine subsystem is then proposed as the source of the fault.

For each proposed faulty component, DRAPHYS propagates the fault through the directed graph of paths of propagation. Nodes in the graph represent components, and links represent dependencies among the components. The links indicate only that one component may affect another; it does not say how soon, or in what direction, or in what amount. DRAPHYS propagates the effect of the proposed fault to other components and checks the real mechanism to see if symptoms have appeared at the predicted places. Propagation halts on any path where the affected component is not yet symptomatic. DRAPHYS retains hypotheses that account for all symptoms, extending the propagation as new symptoms appear. If a hypothesis cannot account for a symptom, the hypothesis is discarded. The concept of propagating the effects of a fault is exactly analogous to tracking the propagation behavior of diseases in a causal network, as done in CASNET [WKAS78]. (CASNET is arguably the first major model-based reasoning program.)

DRAPHYS was evaluated on eight civil transport aircraft accident cases, and correctly diagnosed seven of them. In explaining why DRAPHYS worked so well, Abbott credits two aspects of her approach, both of which are also embodied in MIMIC. The first credit goes to the rapid detection of discrepancies by comparing sensor readings to expected values computed from a numerical simulation model. In contrast to fixed-threshold alarms, this approach yields earlier detection of small deviations. The second credit goes to the model of fault propagation which enables efficient tracking of symptoms against hypothesis-specific expectations.

Although not a particular emphasis of her work, DRAPHYS provides a separate method for diagnosis of well-known, commonly occurring faults. The approach embeds temporal predicates into a rule-based system using the temporal functions of Allen [All84] (e.g., `Starts, Meets, Overlaps,` etc.). For example, one of the rules is triggered when variables EPR, EGT, and Fuelflow are fluctuating simultaneously, and EPR fluctuating is followed immediately by EPR decreasing, and EGT fluctuating is followed immediately by EGT decreasing. This is essentially a fault signature expressed as a sequence of observations, and allows for quick recognition of specific patterns. Although this approach is easy to understand, Abbott points out two specific limitations. First, it is difficult to get knowledge about faults and their propagation behavior at this level of detail. In particular, it is difficult to predict all the different manifestations that a particular fault can exhibit. A second problem is the choice of rule-based representation. When using rules, the entire sequence of symptoms must take place before the rule is satisfied. However, it is sometimes important to identify those fault hypotheses whose initial temporal sequence is satisfied, even if subsequent symptoms have not yet occurred. This capability is inherent in MIMIC's tracking of multiple hypotheses over time, and Abbott's observation supports a basic principle in the design of MIMIC: rather than trying to encode specific symptom patterns, it is better to use simulation to reveal all the possible patterns.

DRAPHYS can be viewed as performing much the same kind of monitoring and diagnosis as MIMIC, albeit at a higher level of abstraction. Both detect discrepancies by comparing readings to predictions from a quantitative model, and both track the evolving symptoms against fault-specific expectations. There are, however, fundamental differences in the models used, leading to significant differences in approach. First, MIMIC uses a semiquantitative model which generates upper and lower bounds on variable values. This simplifies the problem of discrepancy detection — readings are checked to see if they are within the bounds rather than trying to decide if they are "close enough" to some average value. Second, MIMIC uses the semiquantitative model not only to detect initial symptoms but also to verify agreement between readings and specific fault hypotheses. By injecting specific faults into the semiquantitative model, MIMIC can generate expectations that are more detailed than those generated in a graph of propagation paths, and thus more likely to be refuted if incorrect. Third, MIMIC's semiquantitative predictions of (faulty) behavior include time, whereas Draphys' graph of propagation paths does not. MIMIC can use not only the order in which effects occur but also the absolute time as a way to check hypotheses.

2.2.3 MIDAS (Finch, Oyeleye and Kramer)

The Model-Integrated Diagnostic Analysis System (MIDAS) is a program for diagnosing abnormal transient conditions in chemical, refinery, and utility systems [FOK90]. MIDAS is similar to MIMIC in that it continuously monitors the physical plant, updating its hypotheses as new "events" appear. The basic design of MIDAS for on-line diagnosis is depicted in Figure 2.2, as explained below.

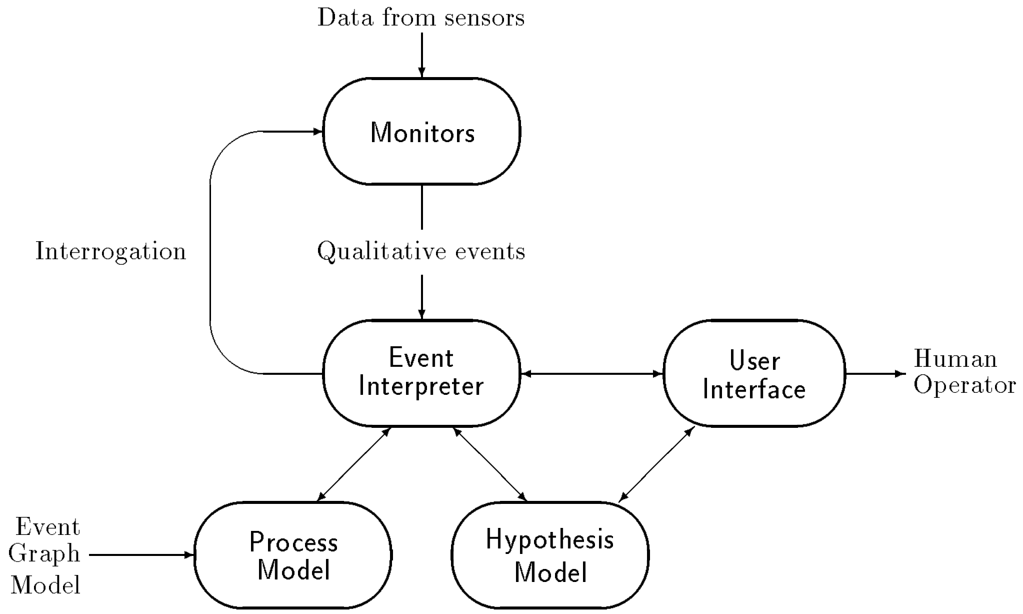


Figure 2.2: Design of MIDAS on-line diagnosis.

Inference in MIDAS centers around the detection of "events" and search within an event model. An *event* is any observable discrete occurrence that carries significant diagnostic information. An event is typically a change in the qualitative state or trend of a process (e.g., level was normal and is now high), or the results of a diagnostic test (e.g., pump test was performed with negative results), or a constraint equation residual (e.g., mass balance was satisfied and is now violated). Events are declared by *monitors*, with a separate monitor for each sensor and each constraint equation. A disturbance detected by any monitor can initiate diagnosis.

Diagnosis depends on the event model. The event model can be viewed as a graph in which nodes represent the different qualitative states of the modeled process variables. For each variable there is a set of nodes, one for each of its qualitative states, only one of which can be active at any given time. For each node which represents an abnormality, there are associated *root causes* (malfunctions). For example, root causes associated with the node "level high" might be outlet blockage, high inflow, and level sensor high bias. Nodes can be connected by precursor/successor links that depict causal relationships between nodes. For example, "level high" may have a successor link to "outflow high" indicating that the former state can lead to the latter state. Conditions are often attached to such causal links to restrict the propagation of a disturbance. For example, although "outlet blockage" is a potential root cause of "level high", it is not a potential cause of "outflow high" and is therefore not supported if "outflow high" is detected. MIDAS' event model is similar in purpose to DRAPHYS' model of paths of fault propagation; both permit new observations to be reconciled with existing hypotheses. MIDAS' event model contains more information in that its links possess conditions under which disturbances can be transmitted.

The event interpreter creates a diagnosis from event observations. For every detected event the interpreter searches through the causal links in the process model for relationships that might exist between the new event and previously detected events. It is assumed that events which can be clustered (i.e., causally related in the event graph) stem from a single root cause. The interpreter then either revises an existing diagnosis or creates a new one.

Interestingly, four specific limitations of MIDAS cited by the authors are addressed by MIMIC:

- MIDAS currently assumes that the process operates in a nominal steady state. It does not simulate a dynamic process model in parallel with the plant to distinguish abnormal transients from normal dynamic behavior. MIMIC is capable of monitoring a process throughout its full range of dynamic behavior.

- MIDAS can explain events but it cannot predict them. Causal links mean "may cause" rather than "will cause". Although MIMIC does not employ a "causal" model, it does use a behavioral model to predict future effects.

- MIDAS assumes that the process remains within a single qualitative regime (with the exception of controller saturation). Malfunctions that add new causalities or reverse signs in the signed directed graph require other methods of diagnosis. In contrast, a process monitored by MIMIC may exhibit [predictable] transitions among different operating modes, and MIMIC may hypothesize and track faults that cause arbitrary changes within components of the mechanism.

- Semiquantitative information about the process can only be used in the quantitative constraints (used by the monitors); they cannot be used in the SDG and therefore cannot strengthen the event model. In MIMIC, semiquantitative models are used not only for detecting discrepancies but also for predicting what happens next, and in what order.

2.2.4 Inc-Diagnose (Ng)

In 1987 Reiter proposed a formal theory of diagnosis from first principles that reasons from system descriptions and observations of system behavior [Rei87]. The algorithm computes all minimal diagnoses of a device, including multiple-fault diagnoses. In contrast to an abductive approach to diagnosis in which the hypothesized faults must imply or explain the symptoms, Reiter's theory requires only that a diagnosis be consistent with the system description and observations; no notion of causality is needed. However, the theory was applied only to diagnosis of digital circuits, which are representative only of physical devices having discrete, persistent states. Reiter's theory is not in conflict with the well-known model-based diagnosis work of researchers such as Davis & Hamscher [DH88] and deKleer & Williams [dKW87]; rather, it lays a stronger theoretical foundation for that work.

In 1990 Ng [Ng90] extended Reiter's algorithm to diagnose dynamic continuous devices of the kind modeled by QSIM. Since QSIM represents continuous behavior as a finite number of discrete qualitative states, it allows the diagnosis problem to be transformed from a continuous one to a discrete one. Ng's algorithm, named INC-DIAGNOSE, makes Reiter's approach incremental in that it permits measurement-taking at different times, intermixed with hypothesis generation. Like Reiter's approach, INC-DIAGNOSE can diagnose multiple faults using only a correct model of the device (it does not use fault models).

Although INC-DIAGNOSE and MIMIC address the same basic problem — diagnosis of continuous dynamic systems — they differ fundamentally in what constitutes a hypothesis. In INC-DIAGNOSE a hypothesis is a set of components believed to be defective. In MIMIC a hypothesis includes the same thing but also specifies a particular fault mode for each device *and* the semiquantitative state of the entire mechanism. This leads to a fundamental difference in the design of the two algorithms: INC-DIAGNOSE uses a model only for consistency-checking of observations; MIMIC uses a model not only for consistency-checking but also for prediction. This leads to several differences in capabilities:

- INC-DIAGNOSE does not verify that measurements taken at time $t+1$ are valid successors of measurements taken at time t, and therefore does not recognize an illegal "jump" in a mechanism's behavior.

- Because INC-DIAGNOSE does not use fault models and does not do prediction, it cannot predict the effects of faults and thus cannot forewarn of undesirable future states.

- Because INC-DIAGNOSE does not predict and track the evolving state of a hypothesis model, it has no information about the possible value of an unobserved state variable other than what it can infer from the latest readings. This weakens its ability to detect inconsistencies.

The problem of multiple-fault diagnosis is treated differently in the two algorithms. INC-DIAGNOSE addresses the more difficult problem where symptoms of two or more faults may appear simultaneously. Although the algorithm is relatively efficient for computing all diagnoses, its performance is still exponential in the worst case. MIMIC makes a simplying assumption that faults (or more precisely, their symptoms) appear one-at-a-time with respect to frequent observations. This allows a restricted form of multiple-fault diagnosis that spreads the exponential work over time.

2.2.5 Kalman Filters

The field of control systems engineering has long been interested in on-line fault detection in dynamic systems. Much of the early work (and much of the current engineering practice) centers around fixed-threshold alarm systems. However, with the declining cost of digital computers and advances in model-based signal processing, the control engineering community has moved toward the idea of simulating a dynamic mathematical model in parallel with the physical system being monitored. The recent book *Fault Diagnosis in Dynamic Systems* [PFC89] is the first multi-authored book on this subject from the control engineering community, and it covers the state of the art in fault diagnosis from several points of view. The book focuses to a large extent on *fault detection and isolation* (FDI) techniques that are based on a dynamic model of a process system — an approach that is similar to MIMIC.

Analytical Redundancy The common theme in all of this work is analytical redundancy — a method of fault detection that uses known analytical relationships among different signals to check for mutual consistency. This approach depends on estimating the values of observed variables using a mathematical model of the system, and then updating the current estimate as new measurement data become available. There are several different forms of model-based algorithms depending on the models used and the manner in which estimates are calculated. For example, there are process model-based algorithms (Kalman filters), statistical model-based algorithms (Box-Jenkins filters, Bayesian filters), statistic-based algorithms (covariance filters), and optimization-based algorithms (gradient filters) [Can86, p. 12]. We will focus our comparison on Kalman filtering since it is based on a deterministic dynamic process model and thus most nearly resembles the type of model used in MIMIC. We begin with a synopsis of the mathematical terminology and then examine what the FDI literature calls "observer-based fault detection". This is followed by a basic introduction to Kalman filters, which is then compared to MIMIC.

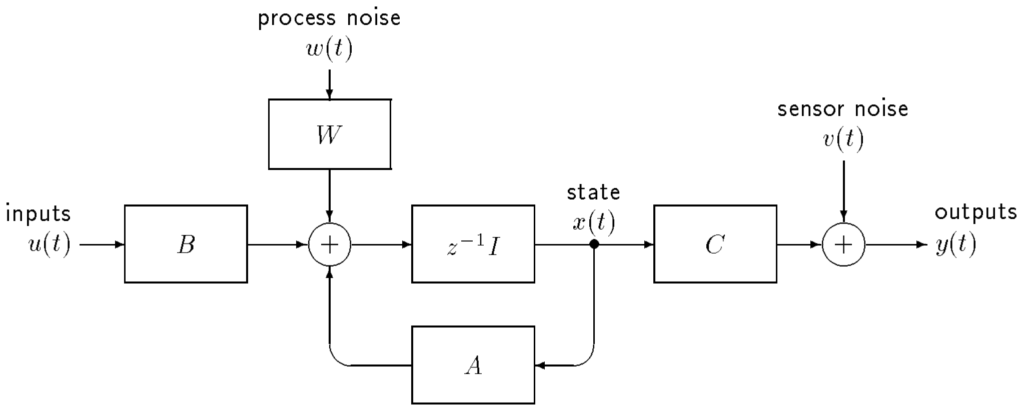
State-Space Models The "modern" approach to system analysis (in contrast to the "classical" approach prevalent prior to World War II) is based on state-space analysis. Sometimes called the state-variable method, it is simply the technique of representing an nth-order differential equation as a set of n first-order equations. The state-space model of a discrete-time process is given by

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1)$$

with the corresponding *output* or *measurement* model as

$$y(t) = C(t)x(t)$$

where x is the N_x-state vector, u is the N_u-input vector, y is the N_y-output vector, A is the $(N_x \times N_x)$-system matrix, B is the $(N_x \times N_u)$-input matrix, and C is the $(N_y \times N_x)$-output



model:

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + W(t-1)w(t-1)$$
$$y(t) = C(t)x(t) + v(t)$$

where:

x, u, and y are the state, input, and output vectors;
A, B, C, and W are appropriately dimensioned matrices;
w and v are zero-mean, white, gaussian noise sequences
with respective covariances $R_{ww}(t)$ and $R_{vv}(t)$;
x(0) is gaussian with mean $\hat{x}(0)$ and covariance $P(0)$.

Figure 2.3: Gauss-Markov model of a discrete process.

matrix. Given a deterministic input $u(t-1)$ and zero-mean, white, random gaussian noise $w(t-1)$, the Gauss-Markov model becomes

$$x(t) = A(t-1)x(t-1) + B(t-1)u(t-1) + W(t-1)w(t-1)$$

where $w \sim N(0, R_{ww})$ and $x(0) \sim N(\hat{x}(0), P(0))$. When the measurement model is included, we have

$$y(t) = C(t)x(t) + v(t)$$

where $v \sim N(0, R_{vv})$. This model is shown in Figure 2.3.

Kalman Filter The basic principle of fault detection and isolation using state estimation is illustrated in Figure 2.4. State estimation is performed by a state observer or Kalman filter. Mathematically, the inconsistency between the actual and expected behavior is expressed as *residuals*. Residuals are quantities that are nominally zero but become nonzero when faults or disturbances are present. While faults can be *detected* from a single residual, fault *isolation* requires a set of residuals [Ger92]. Hence, some decision logic is employed to determine what fault is present given a set of residuals.

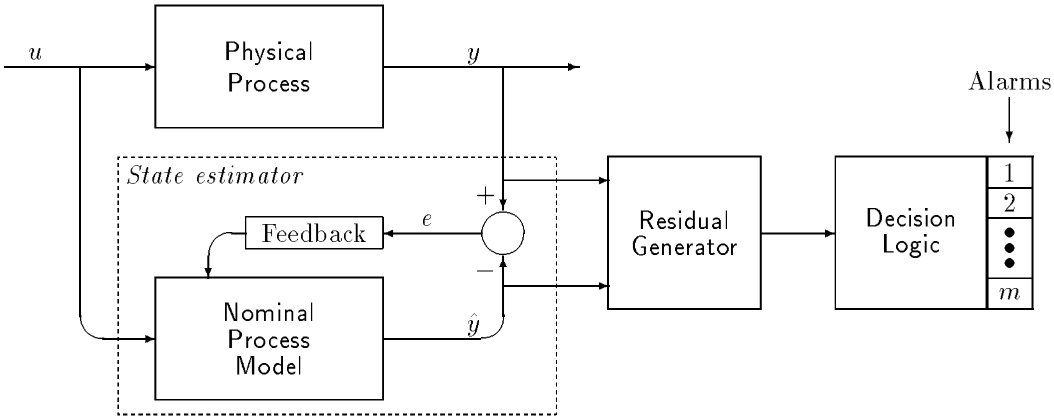


Figure 2.4: Fault detection and isolation using state estimation.

The Kalman filter can be thought of as an algorithm that produces two types of output, given a noisy measurement sequence and associated models. First, it can be thought of as a *state estimator* or *reconstructor*, i.e., it reconstructs estimates of the state $x(t)$ from noisy measurements $y(t)$. In this view it is like an implicit solution of equations since the state is not necessarily available (measurable) directly — the model can be thought of as the means to implicitly extract $x(t)$ from $y(t)$. Second, the Kalman estimator can be thought of as a *measurement filter* which accepts the noisy sequence $\{y(t)\}$ as input and produces a filtered measurement sequence $\{\hat{y}(t|t)\}$. (The notation $\hat{y}(t_2|t_1)$ can be read as "the estimated value of y at time t_2 given the measurements at time t_1".)

The Kalman filter can be described as a predictor-corrector algorithm which alternates between a prediction phase and a correction phase. Candy [Can88, p. 322] gives a succinct description of the algorithm:

> The operation of the Kalman filter algorithm can be viewed as a predictor-corrector algorithm as in standard numerical integration. Referring to the algorithm in Table 2.1, we see the inherent timing in the algorithm. First, suppose we are currently at time t and have not received a measurement $y(t)$ as yet. We have available to us the previous filtered estimate $\hat{x}(t-1|t-1)$ and covariance $\tilde{P}(t-1|t-1)$ and would like to obtain the best estimate of the state based on $[t-1]$ data samples. We are in the "prediction phase" of the algorithm. We use the state-space model to predict the state estimate $\hat{x}(t|t-1)$ and associated error covariance $\tilde{P}(t|t-1)$. Once the prediction based on the model is completed, we then calculate the innovation covariance $R_{ee}(t)$ and Kalman gain $K(t)$. As soon as the measurement at time t becomes available, that is, $y(t)$, then we determine the innovation $e(t)$. Now we enter the "correction phase" of the algorithm. Here we correct or update the state based on the new information in the

Prediction:

$$\hat{x}(t|t-1) = A(t-1)\hat{x}(t-1|t-1) + B(t-1)u(t-1) \qquad \text{(state prediction)}$$

$$\tilde{P}(t|t-1) = A(t-1)\tilde{P}(t-1|t-1)A'(t-1) + W(t-1)R_{ww}(t-1)W'(t-1) \qquad \text{(covariance prediction)}$$

Innovation:

$$e(t) = y(t) - \hat{y}(t|t-1) = y(t) - C(t)\hat{x}(t|t-1) \qquad \text{(innovation)}$$

$$R_{ee}(t) = C(t)\tilde{P}(t|t-1)C'(t) + R_{vv}(t) \qquad \text{(innovation covariance)}$$

Gain:

$$K(t) = \tilde{P}(t|t-1)C'(t)R_{ee}^{-1}(t) \qquad \text{(Kalman gain)}$$

Correction:

$$\hat{x}(t|t) = \hat{x}(t|t-1) + K(t)e(t) \qquad \text{(state correction)}$$

$$\tilde{P}(t|t) = [I - K(t)C(t)]\tilde{P}(t|t-1) \qquad \text{(covariance correction)}$$

Initial Conditions:

$\hat{x}(0|0)$, $\tilde{P}(0|0)$

Table 2.1: Kalman filter algorithm (predictor-corrector form).

measurement—the innovation. The old, or predicted, state estimate $\hat{x}(t|t-1)$ is used to form the filtered, or corrected, state estimate $\hat{x}(t|t)$ and $\tilde{P}(t|t)$. Here we see that the error, or innovation, is the difference between the actual measurement and the predicted measurement $\hat{y}(t|t-1)$. The innovation is weighted by the gain $K(t)$ to correct the old state estimate (predicted) $\hat{x}(t|t-1)$; the associated error covariance is corrected as well. The algorithm then awaits the next measurement at time $t+1$.

The operation of the Kalman estimator is pivoted around the values of the gain matrix K. For small K the estimator "believes" the model; for large K it "believes" the measurement. A Kalman estimator is not functioning properly when the gain becomes small but the measurements still contain information necessary for the estimates. The filter is said to *diverge* under these conditions [Can86, p. 94].

MIMIC and the Kalman filter differ fundamentally in the way that their respective process models handle uncertainty. The semiquantitative model in MIMIC expresses parameter uncertainty directly in the form of conservative ranges, where the true value falls within the range, and expresses functional uncertainty in the form of upper and lower envelope functions, where the true function is bounded by the envelopes. The philosophy here is that uncertainty should be stated explicitly in the model and reflected in the model's predictions. In contrast, the Kalman filter uses a conventional numerical model in which uncertain parameter values are approximated with mean values and uncertain functional relations are approximated with a "close" function. The numerical model gives precise predictions which are then "corrected" to bring them into closer agreement with measurements. Candy summarizes a problem inherent in this approach:

> The process model is usually an approximation to the underlying physical phenomenology, and the model parameters and noise statistics are rarely exact; i.e., the process model used for the estimator differs from the (real) process that generates the measurements. Sometimes the approximation is intentional. For instance, using a reduced-order model in order to decrease computational complexity or linearizing a nonlinear model. It is clear that an imprecise estimator model degrades filter performance .... In fact, these modeling errors or 'model mismatches' may even cause the estimator to diverge. In designing an estimator it is therefore important to evaluate the effect of the approximations made in the selection of the models. [Can86, p. 111]

Kalman filters do not perform diagnosis, of course, but they can determine how well a given process model tracks the observations (by watching $e(t)$). Therefore, the common method is to use a bank of Kalman filters in parallel, each one designed and tuned for a particular fault [FW89, MW89]. The filter yielding the smallest error $e(t)$ represents the best-matching hypothesis. One obvious problem with this approach is the combinatorics — a potentially large number of filters is needed to represent all the types and combinations of faults to be diagnosed. Another problem is that tuning the Kalman estimator is considered an art [Can86, p. 93]. Candy also notes that the Kalman filter "is not a satisfactory (numerically) or efficient algorithm to employ because of the error covariance equations".

Epilogue One thing is very clear from a study of the literature: model-based approaches to monitoring and diagnosis are not unique to the AI community. The basic concept — that of using a model to predict expected behavior, and then using discrepancies between predictions and observations as diagnostic clues — has evolved in two separate communities: the model-based reasoning (MBR) specialty within the AI community and the fault detection & isolation (FDI) specialty within the engineering community. Unfortunately, the two communities are largely unaware of each other, and each could profit from a better understanding of the other's work. Certainly, the MBR community would profit from an understanding of model-based signal processing and the modern approach to system analysis that it is based on, covering topics such as state estimation, parameter estimation, noise filtering, observability, controllability, and stability. Likewise, the FDI community would benefit from an understanding of qualitative constraint models, constraint suspension, assumption-based truth maintenance, and semiquantitative simulation. An extremely valuable contribution to both communities would be a comprehensive article that compares and contrasts the MBR and FDI methods on a sample problem, showing relative strengths and weaknesses of each. This is an action item for future work.

2.2.6 KARDIO (Bratko et al.)

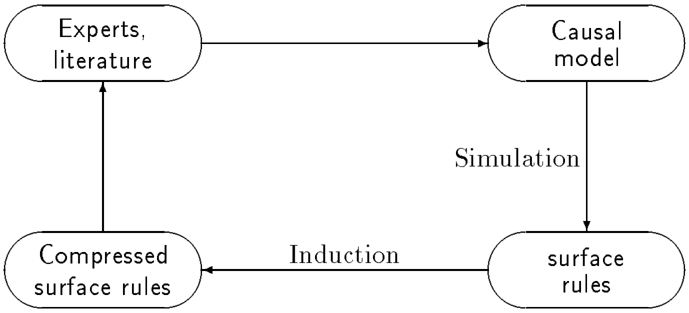


Figure 2.5: Knowledge acquisition cycle used in KARDIO.

KARDIO is a medical expert system for diagnosis of cardiac arrhythmias [BML89]. In contrast with most expert systems whose rules represent heuristics obtained from domain experts, KARDIO's rule base was mechanically generated in a 3-step process: build a qualitative model of the heart's electrical conduction; simulate the model for all interesting combinations of faults; and learn diagnostic rules through induction over the simulation results. The authors (Bratko, Mozetič and Lavrač) expect that the KARDIO "knowledge acquisition cycle" (see Figure 2.5) will become a standard technique in the development of practical expert systems.

It's interesting to note *why* the KARDIO project used a predictive model, because it is for a different reason than is commonly given for model-based diagnostic applications. Most applications, especially those based on engineered mechanisms, use a model partly because it is easily derived from the engineering design and partly because model-based diagnosis overcomes a variety of limitations with symptom-based approaches (see Section 2.1). KARDIO began in 1982 at a time when most diagnostic expert systems were based on experiential knowledge and before Davis had articulated the principles of model-based diagnosis [Dav84]. In the foreword to the KARDIO book, Michie explains why the KARDIO team built a causal model (and then induced diagnostic rules from its predictions):

> That part of existing cardiological knowledge which was explicitly represented in KARDIO was *not* the diagnostic part to be found in texts of clinical practice or in the heads of the authors of such texts. The role of consultant physicians who collaborated in the project was to help the KARDIO team to design the logical model of the heart which was then used as a *de novo* generator of diagnostic rules. *Professor Bratko and his colleagues judged that existing clinical knowledge was not of an explicitness or completeness to support a useful exercise of extraction from specialists* and enhancement in the machine. From the start, therefore, they had no other option but to construct the required corpus of knowledge from scratch — by machine derivation, that is to say, from a compact logical specification. By doing so, Bratko, Lavrač and Mozetič reaped

> the reward of guaranteed completeness and correctness in the finally synthesized expert system. [BML89, Foreword]

There is a lesson here that the model-based reasoning community has perhaps not sufficiently emphasized — that even for systems in which experience-based diagnosis is the norm, it is often easier to acquire and validate a model of the system than to acquire and validate a set of *symptom→fault* associations, particularly where completeness and correctness are concerned. Koton's side-by-side comparison of two expert systems built for the same domain — one using heuristic knowledge, the other using model-based reasoning — offers a good example of this lesson [Kot85].

A fundamental difference in approach between KARDIO and MIMIC lies in the mathematical foundations of the respective simulators. MIMIC builds upon QSIM, a simulation algorithm that implements a qualitative mathematics and guarantees sound predictions of all behaviors consistent with the semi-quantitative model. This soundness eliminates certain sources of false positives during fault detection. Interestingly, the KARDIO team made a deliberate choice to not use an available qualitative modeling technique, as they explain:

> The main feature in the KARDIO approach is the use of logic as the representation formalism. KARDIO is not rooted in any traditional theory used in modeling, such as numerical techniques and differential equations. In comparison with other approaches to qualitative modeling, the main advantage of KARDIO applied to physiology lies in the potential power of the description language used. The model designer has the freedom to choose the most suitable description language and define the laws of the domain in a most natural way. The description language is thus hardly constrained by any traditional mathematical notions. In general, this has of course, the disadvantage that no existing mathematical theory is assumed and automatically available to the model designer. Instead, if such a mathematical theory is useful it has to be explicitly stated in the model. *It really depends on the problem of whether the freedom to choose is more precious than the availability of some established mathematical theory.* The latter is probably more important when the problem is susceptible to some traditional approach, such as differential equations, but in physiology the freedom is probably more valuable. [BML89, p. 49]

The above statement is important but misleading. It is an important warning to modelers to choose a representation that is natural for the domain (wise advice for any project), but it is misleading in that it implies that the modeler must choose between mathematical rigor and expressive power. A more helpful view is that model-based reasoning should be decomposed into two tasks: model building and simulation. The simulator should provide the mathematical rigor, guaranteeing the soundness of all predictions, and the model-building language should make it easy to express what is known about the domain using expressions that are grounded in physics and mathematics rather than *ad hoc* "laws of the domain".

Interestingly, the KARDIO team *did* consider using QSIM, but rejected it because of two important limitations:

> First, it seems to be very difficult to express in QSIM models which are not susceptible to differential equations. This is often the case in medicine and it seems that a QSIM model of the heart that would correspond to the KARDIO model would be extremely complex. Second, even when a QSIM model is derived from corresponding differential equations, the QSIM qualitative simulation algorithm may nondeterministically generate numerous behaviors. Some of the generated behaviors can be justified simply by lack of information in the model. Unfortunately, QSIM also generates unreal behaviors that are not justified by lack of information. [BML89, p. 48].

Although progress has been made on both limitations, it's not clear that the KARDIO team would decide the question any differently today. Their model of cardiac electrical activity involves repetitive processes in time, some of which are asynchronous. While it is possible to model repetitive processes in QSIM, the generated behavior would be large due to all the qualitative distinctions arising from unsynchronized processes. In contrast, KARDIO can assert a simple domain law that says that the sum of two signals, one with regular and one with irregular rhythm, is a signal with irregular rhythm. This kind of abstract symbolic description is suitable for KARDIO's purpose, but is not expressible in QSIM's qualitative differential equations. This highlights the importance of using suitable temporal abstractions, as discussed in section 2.2.7.

The Compilation Debate The KARDIO project was seen as the first clear demonstration that large-scale automatic synthesis of human-type knowledge was technically feasible, fulfilling a goal of the International School for the Synthesis of Expert Knowledge. As such, it was driven by a different research agenda than the other model-based systems described in this chapter. The central approach in KARDIO — that of performing exhaustive simulations of the model and then compressing the predictions into diagnostic rules — has become a topic of debate in the model-based reasoning community. Davis has argued that this approach, which has been employed in numerous other research efforts, does not result in faster diagnostic performance and, more importantly, focuses attention away from the truly important research issue: the creation of a sequence of increasingly approximate models for use by a model-based reasoner [Dav89]. In one of the more spirited debates this author has witnessed, Keller defends the attacks on "knowledge compilation" and argues that it is more fruitful to view associational reasoning and model-based reasoning as opposite endpoints along a spectrum of approaches ranging from more compiled to less compiled [Kel90]. The debate is clearly not settled for there are strongly held views on both sides.

2.2.7 Modeling for Troubleshooting (Hamscher)

Model-based diagnosis, as commonly described in the AI community [DH88], requires models of structure and behavior of the mechanism under study. One might assume that an adequate structural model is simply a complete hierarchical description of subsystems and components, and that an adequate behavioral model is a detailed simulation model that predicts *all* the time-varying events/changes in the mechanism. While such models *do* support model-based diagnosis, they are also part of the problem. As Hamscher notes in a recent article about troubleshooting digital circuits [Ham91], "existing methods for model-based troubleshooting have not previously scaled up to deal with complex digital circuits, in part because *traditional circuit models do not represent aspects of the device that troubleshooters consider important.*" As Hamscher emphasizes, the important thing about developing a model for troubleshooting is not that it uses abstractions to deal with complexity (any representation does that), but that it embodies structural and behavioral abstractions appropriate to troubleshooting.

What sort of abstractions are helpful in troubleshooting? Hamscher identifies eight principles to guide a knowledge engineer in constructing a model that makes troubleshooting feasible. Although Hamscher's research concerns digital circuits (a form of discrete-event dynamic systems), the principles are general enough to apply, in most cases, to the continuous-variable dynamic systems considered in this dissertation. We state the eight principles at the end of this section for interested readers, but here we focus on one issue that may be of particular importance in the future development of MIMIC — the issue of temporal abstractions.

Consider the example of an oscillator in a digital circuit. In a clock-cycle-by-clock-cycle description of the oscillator's output, the output is a repeating sequence of rising edge, stable value, falling edge, and stable value. A more abstract description is the output frequency, and an even more abstract description is simply the attribute "changing" (as opposed to "constant"). These temporally coarse descriptions of behavior hold two advantages: they are easier to generate because the model is simpler, and they are easier to observe in the real mechanism, often providing sufficient diagnostic information to exonerate many components without resort to the more temporally-detailed models. As Hamscher emphasizes:

> *The important property of temporal abstractions is that they sacrifice precision without sacrificing the ability to detect faulty behavior. In troubleshooting the idea is to detect discrepancies between the observed behavior of the real device and an idealized model of it; thus the predictions of interest are those that can be made efficiently from what we have observed and that could be significantly violated if the device were broken.* [Ham91, p. 239]

Temporal abstractions provide considerable leverage in the troubleshooting process, and just as they are important in certain types of digital circuits, they are also likely to be important in certain types of continuous-variable dynamic systems. Currently, QSIM can describe a rhythmic signal in the temporally-detailed form of a time-varying magnitude or in the less-detailed form of a frequency. However, to obtain coarser descriptions such as "regular vs. irregular rhythm" or "changing vs. constant", a different modeling language must be used. The development of such a language to support temporally coarse descriptions of behavior in continuous-variable dynamic systems will enable systems like MIMIC to exploit an important type of abstraction. Ideally, such a language will be built on a firm foundation of qualitative mathematics and will be able to make guarantees of soundness, as QSIM does. This is an important goal for future work.

Eight modeling principles This section quotes Hamscher's eight principles for guiding knowledge engineers in the construction of models intended for troubleshooting. These principles divide into three categories: behavior, structure, and failures. We show the eight principles here to give interested readers a fuller understanding of Hamscher's insights, and we recommend his article for explanations of each.

Modeling of Behavior

- The behavior of components should be represented in terms of features that are easy for the troubleshooter to observe.
- The behavior of components should be represented in terms that are stable over long periods of time or that summarize much activity into a single parameter. This is easiest for a component for which changes on its inputs always results in changes on its outputs.
- A temporally coarse behavior description that only covers part of the behavior of a component is better than not covering any at all.
- A sequential circuit should be encapsulated into a single component to enable the description of its behavior in a temporally coarse way.

Modeling of Structure

- Components in the representation of the physical organization of the circuit should correspond to the possible repairs of the actual device.
- Components in the representation of the functional organization of the circuit should facilitate behavioral abstraction.

Modeling of Failures

- An explicit representation of a given component failure mode should be used if the underlying failure has high likelihood.

- An explicit representation of a given component failure mode should be used if the resulting misbehavior is drastically simpler than the normal behavior of the component.

2.3 Influential Research

2.3.1 Measurement Interpretation

As Forbus explains in a 1986 paper:

> *An unsolved problem in qualitative physics is generating a qualitative understanding of how a physical system is behaving from raw data, especially numerical data taken across time, to reveal changing internal state. Yet providing this ability to "read gauges" is a critical step towards building the next generation of intelligent computer-aided engineering systems . . .* [For86]

This problem, called *measurement interpretation*, arises in MIMIC and other systems that attempt to translate observed behavior (including numerical data) into useful qualitative terms. In this section we briefly review Forbus' theory of "across-time measurement interpretation" (ATMI) and then explain why and how MIMIC takes a considerably different approach to the same problem. This comparison applies equally well to DeCoste's extension to ATMI, named DATMI [DeC90].

ATMI The ATMI theory requires two pieces of input: a total envisionment of the mechanism (a graph of all possible behaviors) and domain-specific criteria for quantizing numerical data into an initial qualitative description. A measurement interpretation problem takes as input a set of measurement sequences, each consisting of a set of measurements for a given variable, totally ordered by the times of the measurements. The output is a set of one or more consistent interpretations of the data expressed as a finite path through the total envisionment. In other words, given the measurements of a mechanism across time, an interpretation is that the mechanism went through a specific sequence of qualitative states S_1, S_2, $\ldots S_n$.

The ATMI theory interprets measurements in a manner analogous to AI models of speech understanding in which the speech signal is partitioned into segments, each of which is explained in terms of phonemes and words, and where grammatical constraints are imposed between the hypothesized words to prune the possible interpretations. In ATMI, the initial signal sequence is partitioned into pieces which are interpreted as possible qualitative states of the mechanism. The envisionment, by supplying information about state transitions, plays the role of grammatical constraints, imposing compatibility conditions between the hypotheses for adjacent partitions.

ATMI performs interpretation with respect to a total envisionment of a model of the mechanism (normally, the fault-free model). In order to perform interpretations of possibly faulty behavior, ATMI would also require total envisionments of every type and combination of fault to be diagnosed. For mechanisms of the scale and complexity found in the process industries, computing the total envisionments is an enormous job. Forbus recognizes this and proposes that the envisionments be pre-computed and preprocessed to provide a set of state tables indexed by the possible values of measurements.

Mimic MIMIC's approach to measurement interpretation differs in two important ways. First, MIMIC is continuously tracking observations against an incremental simulation. Every state S in the tracking set represents a test for new observations — they are either compatible with S or not. If not, tracking simply looks at the near successors of S for a match (this is explained more fully in Chapter 3). MIMIC does not require the massive precomputing of total envisionments, both normal and faulty. Instead, it generates an attainable envisionment, extending the simulation only as needed, and pruning behavior branches as soon as they fail to match observations. In effect, the simulation is focused by the observations, and interpretation is reduced to local search in the unfolding attainable envisionment.

The second difference between MIMIC and ATMI is in the interpretation of individual numerical measurements. In ATMI, a numerical measurement is converted into one or more qualitative values based on pre-specified conversion tables. This quantitative-to-qualitative conversion loses information. In MIMIC, however, there is no such information loss. Instead, the semiquantitative simulation generates numeric ranges for each variable, and the measurement is simply tested for overlap with the predicted range. In fact, there is information *gain* because each new set of observations is unified with the model's predictions to update the current state. Thus, by using the information in previous measurements, MIMIC is able to generate tighter bounds on expected values.

In a sense, the MIMIC and ATMI approaches to measurement interpretation are complementary. ATMI is able to take an arbitrary sequence of measurements as input and produce one or more possible interpretations, having no idea of the state of the mechanism when the first measurement was taken. MIMIC, in contrast, begins with the initial state of the mechanism and so is able to build an interpretation as measurements arrive with less search. Also, because of its diagnostic capabilities, MIMIC is able to build interpretations that jump across models as faults and repairs are diagnosed.

2.3.2 Generate, Test and Debug

In 1987 Simmons and Davis presented a problem-solving paradigm named "Generate, Test and Debug" (GTD) that combines associational rules and causal models, producing a system with both the efficiency of rules and the breadth of problem solving power

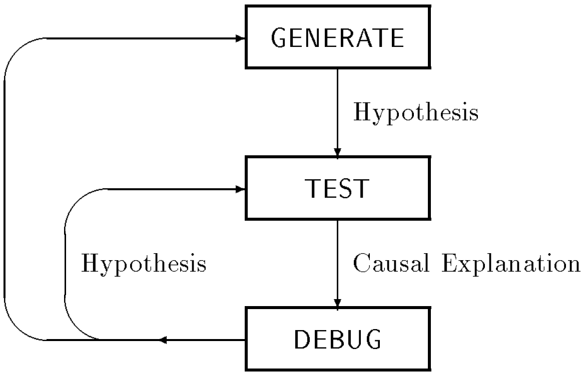


Figure 2.6: Control flow in the GTD paradigm.

of causal models [SD87]. GTD was explored primarily for planning and interpretation tasks. Both tasks are of the general form "given an initial state and a final (goal) state, find a sequence of events which could achieve the final state." Admittedly, this is different than the diagnosis task addressed by MIMIC; we'll explain the influence of GTD on MIMIC following a synopsis of GTD's three stages.

Problem solving in GTD proceeds in three stages, as shown in Figure 2.6:

1. **Generate** — the generator uses associational rules to map from effects to causes. The left-hand side of a rule is a pattern of observable effects and the right-hand side is a sequence of events which could produce those effects. The rules are matched against the final state and the resultant sequences are combined to produce an initial hypothesis — a sequence of events that is hypothesized to achieve the final state.

2. **Test** — the tester tests the hypothesis by using a causal model to simulate the sequence of events. If the test is successful (i.e., the results of the simulation match the final state) then the hypothesis is accepted as a solution. Otherwise, the tester produces a causal explanation for why the hypothesis failed to achieve the final state. It then passes this explanation and the buggy hypothesis to the debugger.

3. **Debug** — the debugger uses the causal explanation from the tester to track down the source of the bugs in the hypothesis. It uses both domain-specific causal models and domain-independent debugging knowledge to suggest modifications which could repair the hypothesis. The modified hypothesis is then submitted to the tester for verification. Alternatively, the debugger has the option to invoke the generator to produce a new hypothesis.

The core idea in GTD is of "debugging almost right plans" whereas in MIMIC it is of "debugging almost right models". The interaction between GTD's tester and debugger is similar to the interaction between MIMIC's discrepancy detector (as tester) and hypothesis generator (as debugger). The discrepancies are used by the hypothesis generator to decide how to modify the model, which is then simulated to see if its predictions are corroborated by future measurements. This is described in more detail in Chapter 3.

2.3.3 STEAMER

STEAMER [HHW84] was a research effort concerned with exploring the use of AI software and hardware technologies in the implementation of intelligent computer-based training systems. The key idea is that of an *interactive inspectable simulation.* For example, STEAMER's animation of a steam propulsion plant allows a student to see processes and conditions that are not visible in the physical system. This is felt to be an important contribution in the student's formation of a mental model. In a warning about increasing automation in the control room, Perrow emphasizes how important it is for an operator to understand his/her system:

> *This computerization has the effect of limiting the options of the operator, however, and does not encourage broader comprehension of the system—a key requirement for intervening in unexpected interactions.* [Per84, p. 122]

STEAMER's main contributions are in the areas of training and human interface. Although these are not the focus of our research, STEAMER is important in that it highlights the importance of visualizing the system, even the parts that aren't visible. In a process monitoring situation where a human operator is the final arbiter, it is very important to promote the operator's understanding of each hypothesis and the associated state of the mechanism. Since MIMIC predicts the values of unseen variables, it can *show* the operator the complete state of the mechanism for each hypothesis.

Chapter 3

The Design of Mimic

Most medical expert systems diagnose by naming a disease rather than coherently describing what is happening in the world (a model) and causally relating states and processes. Perhaps because programs do not structurally simulate pathophysiological processes, researchers have not considered inference in terms of model construction.

William J. Clancey [Cla89]

Clancey's observation above suggests that a diagnostic program should simulate the faulty process under observation, attempting to construct a model that tracks the physical situation. This is, in fact, a succinct description of MIMIC's operation.

This chapter describes the design of MIMIC. The objective here is to present the design in enough detail to enable a rational reconstruction. We proceed from general to specific, starting with the high-level architecture and then elaborating on its components.

3.1 Design Overview

MIMIC is a model-based design for monitoring and diagnosis of continuous-time dynamic systems. Its two basic paradigms are "monitoring as model corroboration" and "diagnosis as model modification". Figure 3.1 presents an abstract view of the MIMIC architecture in which three tasks mediate between the physical system and its models. The three tasks are summarized as follows:

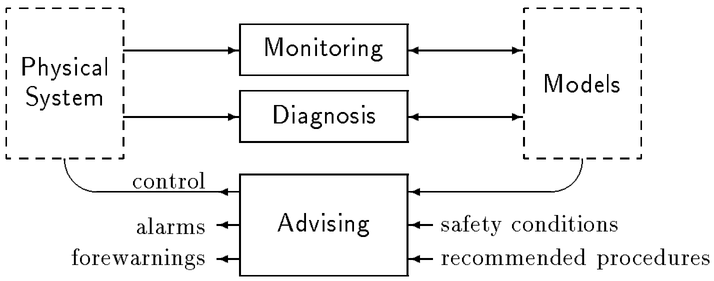


Figure 3.1: Three tasks of an operator advisory system.

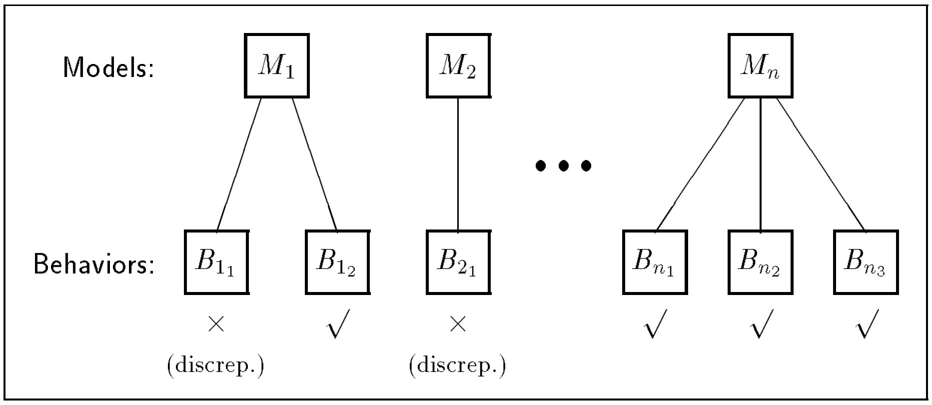


Figure 3.2: Each element of the tracking set is a model plus its tracked behavior(s). A behavior is removed when it fails to match current observations, such as B_{1_1} and B_{2_1}. A model is removed from the set only when it has no remaining behaviors, such as model M_2.

Monitoring The purpose of the monitoring task is twofold: to update the state of the model in synchrony with the mechanism's observed state, and to detect when differences between observations and predictions indicate a fault. This task is driven by the arrival of new observations in a sense-simulate-compare-update cycle. When new observations arrive, the model is simulated up to the time of the observations and observations are compared to predictions. Any discrepancies indicate that the model no longer reflects the mechanism, due to a fault or repair in the mechanism. If there are no discrepancies, the state of the model is updated from the measurements.

Diagnosis The purpose of the diagnosis task is to bring the model back into agreement with the mechanism, and in so doing, identify the fault(s) present in the mechanism. Using discrepancies as clues, this task performs a hypothesize-build-test sequence that creates modifications of the discrepant model, where each modification may add or remove a single fault. These newly hypothesized models are tested for compatibility with the current observations and then injected into the monitoring cycle for subsequent testing as new observations arrive.

Advising The purpose of the advising task is to inform the operator of the current state of diagnosis. Each hypothesis consists of a model (possibly containing faults) and its currently tracked state(s). Hypotheses are ranked by probability, degree-of-match, age, and risk. Ranking by risk (future possible undesirable states) provides an early warning capability that takes advantage of the model's predictive power.

The Tracking Set The set of candidate models maintained by MIMIC is called the *tracking set*. As depicted in Figure 3.2, each element of the tracking set is a model plus its currently

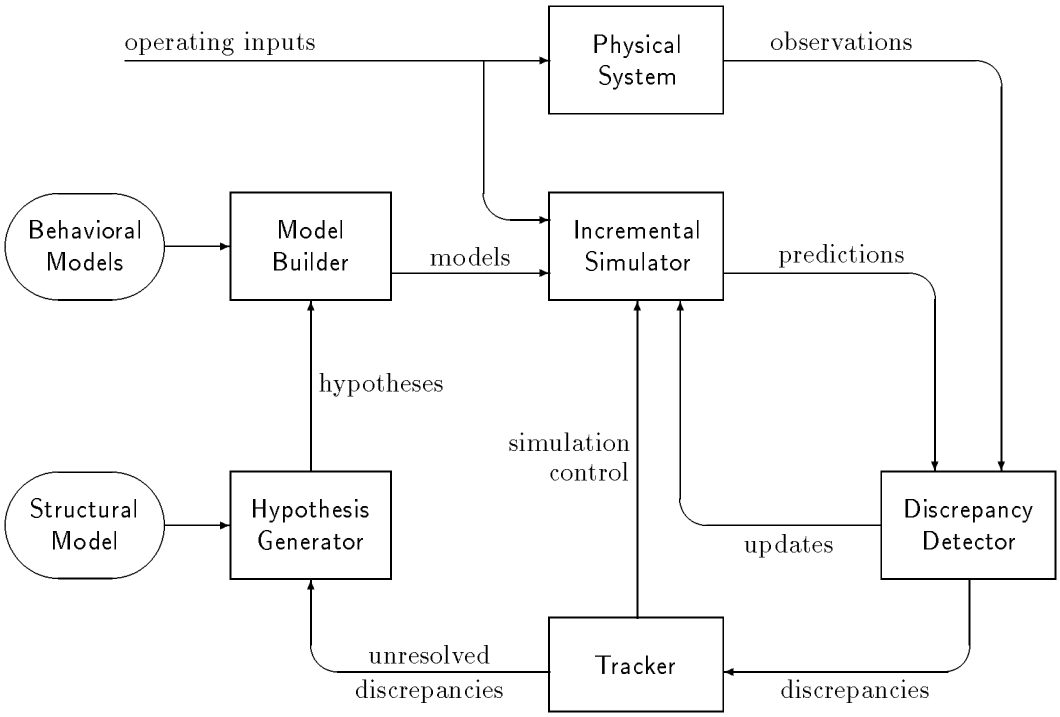


Figure 3.3: Architecture of Mimic. The rectangular boxes represent processing elements and the labeled lines show information flow.

tracked partial behavior(s). Each model is distinguished by the fault hypothesis it embodies. A given model may have more than one distinct behavior being tracked at a given time.

As new readings are obtained and compared to behaviors in the tracking set, some behaviors may be corroborated (denoted with "$\surd$") while others are refuted (denoted with "$\times$"). As long as a model has at least one behavior corroborated up to the present time (such as M_1 and M_n), the model remains in the tracking set with its corroborated behaviors. If all of a model's behaviors are refuted (as with M_2), the model is refuted and removed from the tracking set. The refuted model and its refuted behavior(s) become input to the hypothesis generation function.

Information Flow The flow of information within Mimic can be seen more clearly in the detailed architecture of Figure 3.3. When observations arrive at time t, the incremental simulator advances the simulation of each model in the tracking set to time t. Predictions are then compared to observations, and if no discrepancies are detected, the observations are unified with the predictions and propagated through the model's equations to update the ranges of variables and constants. If a discrepancy is detected, the tracker attempts to resolve it by determining if the observations are compatible with a successor of the current qualitative state. If so, the qualitative state of the model is updated; if not, the unresolved discrepancy triggers hypothesis generation.

The hypothesis generator takes unresolved discrepancies as symptoms of an unknown fault or repair. By dependency tracing through the structural model, suspected components and parameters are identified. For each suspect, single-change hypotheses are formed using a subset of the suspect's operating modes (the subset depends on the type of fault — abrupt or gradual). For each hypothesis, a model is built and then initialized from current observations. If the model initializes successfully, it is added to the tracking set. Failure to initialize indicates a contradiction between model and observations, in which case the model is discarded as an incorrect hypothesis.

Main Loop MIMIC operates in a continuous loop, as shown in Figure 3.4, where each cycle begins with a new set of sensor readings and ends with an updated set of hypotheses. The algorithm operates on the tracking-set, performing three basic actions: (1) it updates the state of candidates that are consistent with the readings; (2) it removes candidates that are inconsistent with the readings; and (3) it creates new candidates based on the discrepancies found in inconsistent candidates.

The remainder of this chapter describes each element of the design in more detail. The first two sections — modeling and simulation — describe the foundations that everything else is built upon. The next three sections — monitoring, diagnosis and advising — correspond to the three main tasks shown in Figure 3.1. The last section describes how complexity is controlled.

3.2 Modeling

MIMIC depends on two distinct types of models: a structural model representing components and connections, and a behavioral model that predicts possible behaviors given a fault hypothesis. This section describes both types of models using a two-tank cascade as an example (see Figure 3.5).

3.2.1 Structural Model

The structure of a mechanism is its components and connections. Figure 3.6 shows the structure of the two-tank cascade. The structural information is used only during diagnosis to trace "upstream" from the site of a discrepancy along the paths of interaction to the components and parameters whose malfunction could have caused the discrepancy. As such, the structural model must contain components, parameters, connections, and the direction of connections.

The structural model used by MIMIC is shown in Figure 3.7. This model is a declarative representation of the preceding structure diagram. The model has three main

Read sensors:

```
    Obtain next time-stamped set of sensor readings for time t.
```

Track candidates:

```
    For each model in tracking-set:
        Simulate ahead to time t,
        Test for discrepancies between readings and predictions.
        If no discrepancies, update state from readings.
    Return two sets:  retained-candidates and rejected-candidates.
```

Generate hypotheses:

```
    Given the discrepancies in rejected-candidates,
    perform dependency tracing to create new single-change
    hypotheses in hypothesized-candidates.
```

Test hypotheses:

```
    Attempt to initialize each member of hypothesized-candidates.
    Successful members are placed in new-candidates.
```

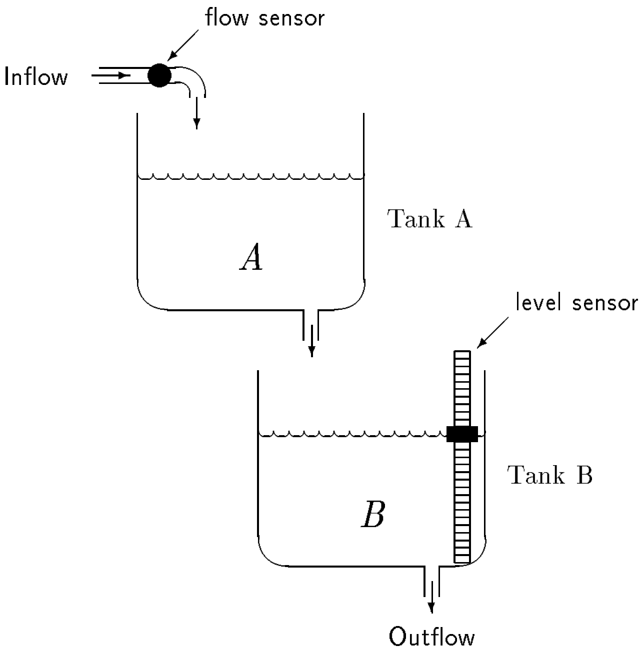
Display candidates:

```
    Display candidates ranked by probability, similarity, age,
    and risk.
```

Update tracking set:

```
    tracking-set ← new-candidates ∪ retained-candidates
    Go to first step.
```

Figure 3.4: Main loop of MIMIC.



Differential equations:

$$A' = inflow - f(A)$$
$$B' = f(A) - g(B)$$

Figure 3.5: Two-tank cascade.

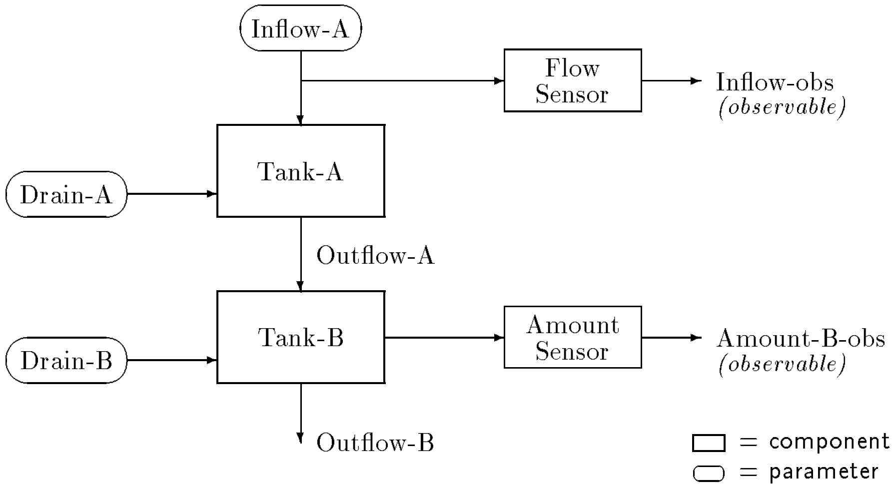


Figure 3.6: Component-connection graph of the two-tank cascade. Rectangles represent diagnosable components/subsystems, ovals represent parameters (constants), and directed links represent connections and their direction of flow.

Component definitions:	*Parameter to connection mappings.*

(COMPONENTS
`(INFLOW-SENSOR`
`(flow-in   input  icon1)`
`(measured  output i-obs))`
`(TANK-A`
`(inlet     input  icon1)`
`(outlet    2-way  tocon1)`
`(drainrate input  dr1))`
`(TANK-B`
`(inlet     input  tocon1)`
`(outlet    2-way  tocon2)`
`(drainrate input  dr2)`
`(amount    output acon))`
`(AMOUNT-SENSOR`
`(amount-in input  acon)`
`(measured  output a-obs)))`
	(PARAMETERS
	`(Inflow-A     icon1)`
	`(Drain-A      dr1)`
	`(Drain-B      dr2))`
	
	Variable to connection mappings.
	
	(VARIABLES
	`(Amount-B-obs a-obs)`
	`(Inflow-obs   i-obs))`

Figure 3.7: Structural description of the two-tank cascade. This is a declarative representation of the component-connection graph in the previous figure.

clauses. The `VARIABLES` clause contains entries for each of the observable variables, typically sensor outputs. The `PARAMETERS` clause contains entries for each of the constants that may be identified as a suspect during diagnosis. Likewise, the `COMPONENTS` clause contains entries for each of the components/subsystems that may be identified as a suspect during diagnosis.

In a component-connection representation of a mechanism it is assumed that *all* interactions among components take place through explicit connections among the *terminals* of the components. Each terminal of a component is described with a 3-tuple consisting of:

(*terminal-name*   *direction*   *connection-name*)

The *direction* of a terminal must be `input`, `output`, or `2-way`. To be called an `input` terminal, the component must not have any direct effect on the quantities transmitted into that terminal. For example, the input of an infrared sensor does not affect the sources of infrared radiation that it measures. To be called an `output` terminal, the quantities transmitted by the component to its terminal must not be directly affected by whatever that terminal is connected to. For example, the luminance at the surface of a light bulb's globe is not affected by the air or fluid surrounding the globe. When a terminal is called `2-way`, it means that some or all of the quantities carried through that terminal can be affected not only by the component but also by what it is connected to it. For example, the amount of flow out of a water tank is affected by the downstream resistance of the piping system it is connected to. These directions-of-effect are used in the dependency tracing algorithm described in section 3.5.1.

The other two elements of the 3-tuple — terminal name and connection name — are simply labels. Terminal names (such as `inlet` and `outlet`) exist only for human understanding and play no role in dependency tracing. Connection names are arbitrary tags which, when repeated in another clause, designate a connection between the associated terminals and/or variables and/or parameters.

3.2.2 Behavioral Model

The purpose of a behavioral model is to predict the possible behavior(s) of a mechanism given a starting state and a fault hypothesis. Further, the predictions are to be based on "first principles", i.e., physical laws such as conservation of energy, mass balance, Ohm's law, etc. For the class of mechanisms considered in this report, the model is a deterministic, continuous-time, dynamic model based on an abstraction of ordinary differential equations. The qualitative differential equation (QDE) representation that we use is based on Kuipers' formalism for qualitative simulation [Kui86] and further extended by Kuipers and Berleant for simulation with incomplete quantitative knowledge [KB88]. This section summarizes the QDE representation.

Figure 3.8 shows a behavioral model for a two-tank cascade where there is a constant inflow into the top of tank-A and an outflow from the bottom of tank-B. The `quantity-spaces` clause defines the variables of the model and the ordered landmark values in each quantity space. The `constraints` clause contains the equations of the model, representing the physical laws of the domain. These two clauses define the *qualitative* model, which can be simulated using only the symbolic non-numeric information shown.

However, the model can be strengthened with partial quantitative knowledge in two additional clauses. First, the `initial-ranges` clause defines numeric ranges for landmark values, allowing the modeler to specify the lower and upper bounds for imprecisely known constants. Second, the `m-envelopes` clause specifies envelope functions for monotonic (`M+` and `M-`) function constraints. This lets the modeler express uncertainty about the precise function by bounding it with upper and lower "envelope" functions. In the example shown, the envelope functions express the basic square root relationship for drains, with uncertainty about the precise value of a factor.

Where, exactly, does the partial quantitative knowledge come from? It is a form of domain knowledge which is sometimes given explicitly in the domain and sometimes requires the judgement of a domain expert. As an example of the former, electrical resistors are labeled as having a nominal value with a tolerance of, say, ± 2 percent. Thus, the initial range for the resistor's nominal value is easily provided. As an example of the latter, in the human cardiovascular system the mathematical relation between central venous pressure

```
(define-QDE Two-Tank-Cascade
  (quantity-spaces
    (Inflow-A    (0 normal inf))
    (Amount-A    (0 full))
    (Outflow-A   (0 max))
    (Netflow-A   (minf 0 inf))
    (Amount-B    (0 full))
    (Outflow-B   (0 max))
    (Netflow-B   (minf 0 inf))
    (Drain-A     (0 vlo lo normal))
    (Drain-B     (0 vlo lo normal)))
  (constraints
    ((M+   Amount-A Outflow-A)   (full max))
    ((ADD  Outflow-A Netflow-A Inflow-A))
    ((D/DT Amount-A Netflow-A))
    ((M+   Amount-B Outflow-B)   (full max))
    ((ADD  Outflow-B Netflow-B Outflow-A))     ; Outflow-A = Inflow-B
    ((D/DT Amount-B Netflow-B)))
  (independent   Inflow-A Drain-A Drain-B)
  (history       Amount-A Amount-B)
  (unreachable-values
    (Netflow-A minf inf)  (Netflow-B minf inf)  (Inflow-A inf))
  (m-envelopes
     ((M+ Amount-A Outflow-A)
        (upper-envelope (lambda (x) (* (ub 'Drain-A) (expt x 0.5))))
        (lower-envelope (lambda (x) (* (lb 'Drain-A) (expt x 0.5))))
        (upper-inverse  (lambda (y) (expt (/ y (ub 'Drain-A)) 2)))
        (lower-inverse  (lambda (y) (expt (/ y (lb 'Drain-A)) 2))))
     ((M+ Amount-B Outflow-B)
        (upper-envelope (lambda (x) (* (ub 'Drain-B) (expt x 0.5))))
        (lower-envelope (lambda (x) (* (lb 'Drain-B) (expt x 0.5))))
        (upper-inverse  (lambda (y) (expt (/ y (ub 'Drain-B)) 2)))
        (lower-inverse  (lambda (y) (expt (/ y (lb 'Drain-B)) 2)))))
  (initial-ranges
    ((Time     t0)      (0    0))
    ((Inflow-A normal)  (8.0  8.1))
    ((Amount-A full)    (45   46))     ((Amount-B full)    (45   46))
    ((Drain-A  normal)  (1.40 1.43))   ((Drain-B  normal)  (1.40 1.43))
    ((Drain-A  lo)      (0.70 1.40))   ((Drain-B  lo)      (0.70 1.40))
    ((Drain-A  vlo)     (0    0.70))   ((Drain-B  vlo)     (0    0.70))))
```

Figure 3.8: Semiquantitative model of a two-tank cascade. Incomplete quantitative knowledge is expressed in the form of ranges for landmark values, in `initial-ranges`, and envelope functions for monotonic function constraints, in `m-envelopes`.

and mean pulmonary arterial pressure is not precisely defined. However, a cardiologist can provide reasonable bounds on this relation.

3.2.3 Modeling Faults

There are two separate properties of faults that are important in MIMIC: how the fault is represented in the model, and speed of onset of the fault. We examine each property below.

Parameter Faults vs. Mode Faults There are two basic ways of representing a fault in a model: by changing the value of a parameter (such as changing the flow resistance of drain because it has become clogged), or by changing the equations of a component (such as changing a heater function that relates voltage input to heat output when the heater burns out). In this section we will show how such faults are represented in the behavioral model. Later, in section 3.5, we will see how such faults are hypothesized.

For faults that are represented as abnormal parameter values, the modeler need only add appropriate landmarks in the parameter's quantity space and corresponding ranges in the initial-ranges clause. An example of this is shown in Figure 3.8, where parameter `Drain-A` can have one of four landmark values: `0`, `vlo`, `lo`, or `normal`. Numeric ranges for these values are defined in `initial-ranges`. Parameters are represented as independent variables in the model, meaning that they are constants. Once a parameter's value is set, it stays that way unless MIMIC changes it during diagnosis.

For faults that change the set of equations in the model, we use the simple idea that the appropriate set of equations for a component depends on the component's operating mode. For example, the mode of an electric heating element may be `normal` or `burned-out`. In the former case it dissipates heat proportional to the square of the voltage applied, but in the latter case it dissipates no heat, regardless of the voltage. This difference in behavior is modeled by having two different sets of equations, only one of which can be active at a time, as shown below in a fragment of a `constraints` clause:

```
(mode (heater normal)
   ((MULT V V V-SQUARED))
   ((MULT HEAT R V-SQUARED)))
(mode (heater burned-out)
   ((ZERO-STD HEAT)))
```

In this example, `heater` is called a *mode* variable, and its value determines which set of equations govern the behavior of the component. Like parameters, mode variables are constants that can be changed during diagnosis. Although we don't explore it in this report, modes serve two other useful purposes: (1) they can be used to represent normal transitions within a component, such as a thermostat that changes between its *on* and *off* modes, and (2) they can be used by a component to predict its own fault, such as a heater that burns out when its power dissipation exceeds a threshold.

Abrupt Faults vs. Gradual Faults MIMIC partitions faults according to their speed-of-onset as *abrupt* or *gradual*. Abrupt faults cover discrete events (such as heater burn-out) and failure of normal behavior transitions (such as a thermostat that fails to turn off after the temperature exceeds an upper limit). Gradual faults cover slowly developing abnormalities caused by aging, drift, contamination, etc (such as a pipe whose flow resistance gradually rises due to lime deposits). Abrupt faults may be modeled as either parameter faults or mode faults, but gradual faults are ordinarily modeled as parameter faults. The distinction between abrupt versus gradual is important during hypothesis generation, as explained below.

Suppose we are modeling an electric heating element in a water heater. The heating element can exhibit two different abrupt faults — short-circuit and open-circuit — which are modeled as mode faults that change the equations relating voltage, current, and heat output. If the heater becomes a suspect during diagnosis, it is proper to hypothesize *all* of its abrupt faults since such faults can happen at any time, regardless of the current operating mode.

Now suppose we are modeling a pipe which is known from experience to accumulate lime deposits over time, even to the point of becoming completely obstructed. We model three levels of severity of this gradual fault by giving the flow-resistance parameter four possible values: `normal`, `high`, `very-high`, and `infinity`. Each of these parameter values will have an associated numeric range specified in the `initial-ranges` clause. If the pipe is operating in the `normal` range but becomes a suspect during diagnosis, then the only appropriate hypothesis is `high`. Because of the fault's gradual nature, `very-high` and `infinity` are not yet valid hypotheses. Thus, the benefit of distinguishing gradual faults from abrupt faults is that fewer hypotheses are generated during diagnosis.

3.3 Simulation

The type of simulation used in MIMIC enables several of the benefits described earlier in section 1.4. Though it is called semiquantitative simulation, a more descriptive name is qualitative-quantitative simulation. The simulation is fundamentally qualitative, but strengthened with partial quantitative knowledge in the form of numeric ranges for landmark values and envelope functions for monotonic function constraints. The resulting simulation combines a key advantage of qualitative simulation — prediction of all behaviors consistent with the partial knowledge — with the ability to use incomplete quantitative knowledge to refine the predictions.

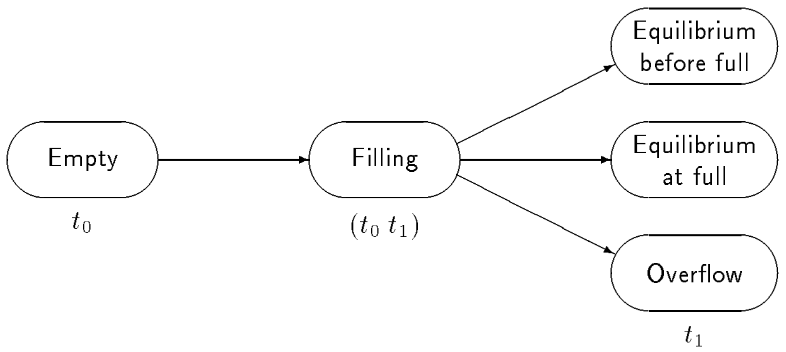


Figure 3.9: Branching-time description of a bathtub's possible behaviors. In contrast to conventional numerical simulation, semiquantitative simulation reveals all behaviors consistent with incomplete knowledge of the mechanism.

This section describes semiquantitative simulation from a user's point of view, focusing on the simulation output and the inherent capabilities of the method. Readers interested in a more complete description of these simulation methods should consult the papers by Kuipers on QSIM [Kui86], by Kuipers and Berleant on Q2 [KB88], and by Kay on dynamic envelopes [Kay91].

3.3.1 Qualitative-Quantitative Simulation

A *behavior*, in QSIM parlance, is a sequence of states alternating between states that represent a point in time and states that represent an interval in time. Point states are created when qualitative distinctions arise during simulation, such as a quantity that reaches a landmark value or changes from increasing to steady. Interval states describe behavior between successive point states; the duration of an interval may range from very short to very long.

Prediction of All Valid Behaviors Two properties of qualitative-quantitative behavior are important in MIMIC. On the qualitative side, the simulation yields a "behavior tree" which is a branching-time description of the possible behaviors of the mechanism. Figure 3.9 shows a simple example of this for a bathtub filling from empty. Initially, the bathtub is empty at time t_0, then begins filling in the time interval $(t_0\ t_1)$. Based on the available information, the bathtub may end in one of three final states: equilibrium before full, equilibrium at full, or overflow. The key feature here is that the branching-time description of behavior *reveals all of the qualitatively distinct behaviors consistent with the available information.* This stands in contrast to conventional numerical simulation which replaces uncertainty with approximations and then generates a single, numerically-precise behavior. The precision is seductive, but it fails to provide a comprehensive description of the behavior space.

The branching-time behavior description is important to MIMIC in two ways. Monitoring in general is susceptible to a "missing prediction error" in which a model is refuted because observations fail to match predicted behavior. The error arises because conventional numerical simulation generates a single behavior even though other behaviors may be possible. This error can cause both false positives and false negatives during discrepancy-detection (false positives when matching against the normal model and false negatives when matching against a fault model). Semiquantitative simulation eliminates this source of error because it guarantees that all valid behaviors are predicted.

The second way that sound behavior prediction is important is in forewarning. When MIMIC is monitoring a fault model, it is important to be able to look ahead in time to see if any undesirable states are imminent. Again, the semiquantitative simulation ensures that all possible futures are predicted, so MIMIC can guarantee that any potential near-future problems are reported to the operator.

Use of Incomplete Quantitative Knowledge On the quantitative side, qualitative-quantitative simulation takes advantage of incomplete quantitative knowledge to constrain the simulation, thereby eliminating some behaviors, and yielding numeric range predictions for every variable. Figure 3.10 shows an example of this for a two-tank cascade, filling from empty. Every landmark value has an associated range. Importantly, the ranges are guaranteed to bound the valid possible values. This allows sensor readings to be compared directly to predicted ranges. This eliminates the "approximate matching problem" that occurs with conventional numerical models since they generate a precise value, thus requiring a decision as to whether the reading is close enough to the predicted value.

3.3.2 Feedback Loops

Feedback is common in natural systems, but its presence has been problematic for many diagnostic systems. As Widman and Loparo note in the introductory chapter of a book on AI, simulation and modeling [WLN89]:

> *Expert reasoning with symbolic models has not yet been widely used despite its evident usefulness. The obstacles lie in two general types of models that are frequently required to describe real-world physical systems: continuous models containing interacting feedback loops, and discrete stochastic models containing interacting probability distributions (conditional dependencies).*

MIMIC works with the first type of model (continuous models containing feedback), so it's important to show how semiquantitative simulation handles feedback. We demonstrate this with an example of negative feedback in an amplifier, as shown in Figure 3.11. For a linear amplifier and linear feedback, the input-output relation can be solved

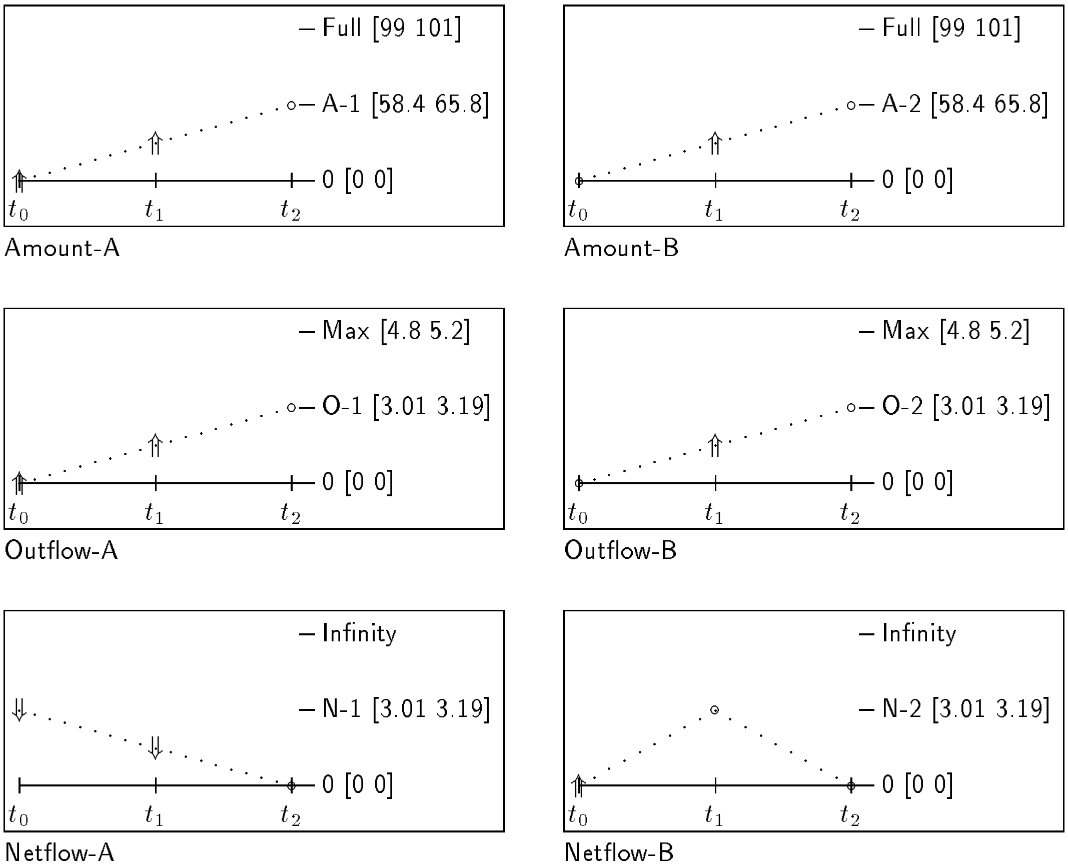
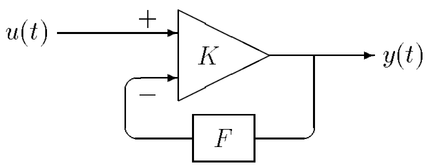


Figure 3.10: Semi-quantitative behavior of a two-tank cascade filling from empty. Time point t_0 represents the starting time and t_2 represents the time when equilibrium is reached. Time t_1 identifies the time at which Netflow-B changes from increasing to decreasing. The ranges associated with each landmark value, such as A-1 [58.4 65.8], are guaranteed to bound the correct value.

K	=	amplifier gain	= [100 101]
F	=	feedback level	= [0.10 0.11]
u	=	input signal	= [3.0 3.1]

Amplifier with feedback



Equation (linear case only):

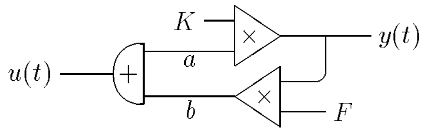
$$y(t) = K(u(t) - Fy(t))$$

$$y(t) = \frac{Ku(t)}{(1+KF)}$$

By analytical solution:

$y = [25.0 \;\; 28.2]$

Constraint model



Constraint equations:

```
(ADD a b u)
(MULT K a y)
(MULT F y b)
```

By constraint satisfaction:

$y = [24.7 \;\; 28.6]$

Figure 3.11: Amplifier with feedback. The amplifier with linear gain K and linear feedback F can be solved analytically, as shown. The semiquantitative constraint model is solved through constraint-satisfaction, yielding a solution guaranteed to bound the correct value. Further, the constraint method works for non-linear models.

analytically: $y(t) = Ku(t)/(1 + KF)$. In general, though, we cannot assume that a mechanism is linear and that an analytical solution exists. QSIM/Q2 does not require an analytical solution. Rather, it solves the problem through constraint satisfaction, propagating ranges through the equations until converging on a final value. In the example shown, given ranges for u, K and F, Q2 converges on a final range for y after eight iterations. Although this technique requires more computation than using an analytic solution, it works equally well for nonlinear systems and for multiple interacting feedback loops.

3.3.3 State-Insertion for Measurements

The qualitative behavior tree of the bathtub, as shown in Figure 3.9, provides a complete map of expected behavior, but it is a very imprecise map. The Filling state, for example, covers the interval of time from the moment when filling begins until the moment that equilibrium is reached. Let's assume that the predicted value for the amount of water in the tub at equilibrium is the range [47 49] liters. Therefore, the predicted value for the amount of water in the tub during filling is [0 49]. If we are monitoring the amount of water during filling, then any measurement in the range [0 49] will be accepted. Obviously, this is as an extremely crude check on measurements and is not very useful in detecting subtle faults.

Time-Point Interpolation The basic problem here is that measurements represents an instant in time, but the interval-state covering that instant covers a potentially long interval of time, so its predictions are imprecise. To overcome this problem we adopt the technique of time-point interpolation from Berleant's dissertation [Ber91, p. 40–50]. By inserting a time-point state into an interval-state (thus dividing it into two interval states) and specifying a time value for the point-state, the Q2 range propagator can significantly tighten the predictions, not only for the newly inserted point-state but also for all succeeding point-states (and for preceding point-states, by propagation). Berleant uses the technique to progressively refine a simulation to some desired precision by adaptively inserting states into wide time intervals.

MIMIC uses time-point interpolation in a slightly different way, in behalf of measurement interpretation. By inserting a point-state for each measurement instant, Q2 is able to provide *much* more precise predictions at each measurement time. This results in much stronger tests of measurements, thus enabling detection of subtle faults. Figure 3.12 shows an example of state insertion for the first five measurements of the two-tank cascade. The right-hand side shows the predicted ranges for `Amount-B` at each measurement instant, providing a strong test for those measurements. As we will see later in section 3.4.3, each inserted state also enables an "analytical test" wherein Q2 checks the mutual consistency of the measurements, assumptions, and model equations.

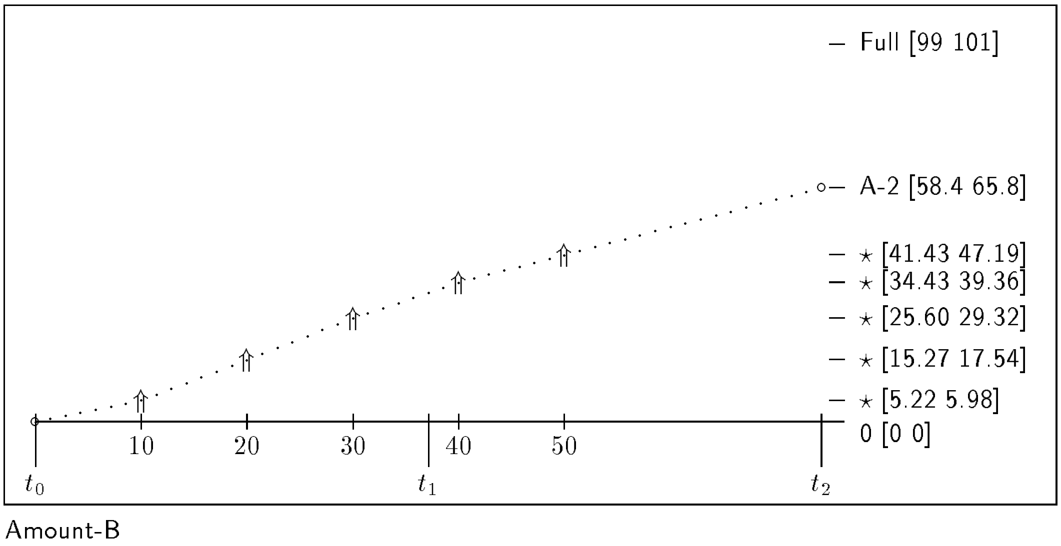


Figure 3.12: State-insertion for measurements. Initially, only the qualitative behavior is known, designated by the qualitative time-point states at t_0, t_1, t_2, and the interval states between them. As new measurements arrive at times 10, 20, 30, 40, and 50, "measurement states" are created for those time points and inserted into the behavior to record the predictions and observations at those instants (the ranges marked with "$\star$" are predictions for those instants). This progressive discretization of the qualitative behavior tightens the predictions for future measurements.

Two Kinds of Time Figure 3.12 illustrates a novel aspect of MIMIC's operation — it reconciles two kinds of time: qualitative and quantitative. Qualitative time is associated with the qualitatively distinct states generated by semiquantitative simulation. These states reveal the space of possible behaviors but provide only weak predictions about actual time-of-occurrence of each event or the duration between successive events. Quantitative time is associated with the passage of measurable time and is therefore associated with predictions and observations for specific instants in time.

In performing time-point interpolation, MIMIC must insert the measurement state into the appropriate qualitative time-interval (or on top of a qualitative time-point). For example, given the measurement at time 40 (see Figure 3.12), should the measurement state be inserted in the time-interval $(t_0\ t_1)$ or on the time-point t_1 or in the interval $(t_1\ t_2)$ or on t_2? The simple answer is that the measurement state is inserted in the first qualitative state that it is *consistent* with, where "consistent" means that there are no detectable discrepancies. This is covered in more detail in section 3.4.

The value of having two kinds of time will become apparent later, but the key benefits can be summarized here. Simulating in quantitative time is essential to generating reasonably precise predictions for testing measurements. Simulating in qualitative time efficiently reveals the space of possible behaviors, and is essential in forewarning the operator of the possible consequences of a fault.

3.3.4 Dynamic Envelopes

MIMIC depends on semiquantitative simulation for two things: prediction of all possible behaviors, and prediction of variable values at each measurement instant. In the latter category we want the most precise predictions consistent with the available knowledge. As we saw in the preceding section, MIMIC uses time-point interpolation with Q2 to obtain more precise predictions at measurement instants. However, range interval propagation in Q2 is still weak over time intervals, and the time between successive measurements can be arbitrarily long. Thus, Q2's predictions at measurement instants may still lack the desired precision.

To overcome the weaknesses in range interval propagation, MIMIC employs a second semiquantitative simulation technique called "dynamic envelopes", developed by Kay [Kay91]. This method replaces the use of the mean value theorem in Q2's propagation with explicit integration of a pair of bounding ordinary differential equations. These bounding ODEs are derived from the QDE and numerically integrated, yielding upper and lower bounds for state values. The method is implemented in a program named NSIM.

MIMIC uses the dynamic envelope method in the following way. When measurements arrive for time t, the bounding ODEs are simulated up to time t. NSIM's predicted ranges for the state variables are then intersected with Q2's predicted ranges and the results are propagated through the measurement state. This ensures that the measurement state

Sense:
```
Obtain sensor readings for time t.
```
Simulate:
```
Advance simulation by inserting state for time t.
```
Compare:
```
Compare readings to predictions.
If discrepancy detected, exit.
```
Update:
```
Update inserted state from readings.
t ← t + 1
Go to first step.
```

Figure 3.13: The Monitoring Cycle. This cycle is applied to every behavior in the tracking set.

has the best available predictions, not only for state variables but also for all other variables (some of which are measured variables).

3.3.5 Pruned Envisionment

QSIM ordinarily generates what is called an *attainable envisionment*, i.e., the set of all possible behaviors attainable from an initial state. This simulation is potentially time-consuming due to the large number of behaviors that result with some models. MIMIC, however, does *not* generate an attainable envisionment. Rather, it simulates incrementally, starting with the initial state and simulating only enough of the behavior to track the current measurements. As observations refute some behaviors, the behavior tree is effectively pruned at that point. The effect, over time, is that MIMIC generates a "pruned envisionment" guided by the measurements, thus greatly reducing the amount of simulation required.

3.4 Monitoring

The purpose of the monitoring task is fault detection. In MIMIC, monitoring consists of a sense-simulate-compare-update cycle, as shown in Figure 3.13. This cycle is applied to every behavior in the tracking set. The end result is that the behavior is either updated and corroborated by the measurements or else it is refuted, with the discrepancies serving as input to the diagnosis task.

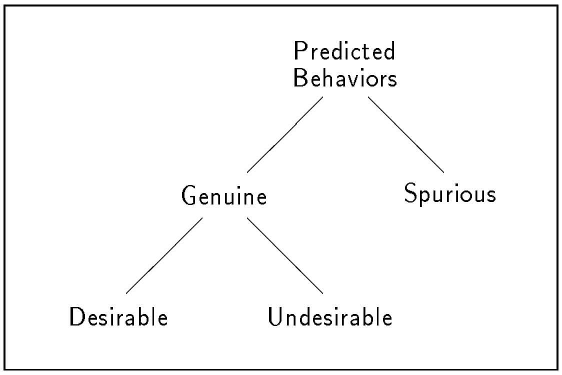


Figure 3.14: Classification of behaviors. The monitor wants to test observations against *desirable* behaviors, but QSIM generates the larger set of *predicted* behaviors, consisting of *genuine* and *spurious* behaviors. Genuine behaviors may include *undesirable* behaviors which, in MIMIC, are distinguished from desirable behaviors by domain-specific criteria represented in "warning predicates". Spurious behaviors may arise due to inadequate physical knowledge in the model or to current limitations of qualitative simulation; most are suppressed through QSIM's global filters.

3.4.1 Monitoring Model

Earlier in section 3.2 we stated that MIMIC requires two types of models — structural and behavioral — but in truth it also requires a third type: the monitoring model. Recall that the behavioral model is used during monitoring to compare observations to predictions. One might assume that the fault-free behavioral model reveals exactly and only the acceptable behaviors of the mechanism, but this is wrong on two counts, as shown in Figure 3.14. First, the behavioral model predicts all possible behaviors of the mechanism, whether or not they are *desirable* behaviors for normal operation. For example, it is predictable for a bathtub to overflow if a large enough inflow is sustained for long enough, but overflow is an undesirable outcome that should produce an alarm. Second, qualitative simulation can produce spurious behaviors that do not correspond to the solution of any ODEs covered by the QDE. If a malfunction exhibits behavior that corresponds to a spurious behavior of the normal model, it will not be recognized as a discrepancy.

As Lackinger notes in his thesis [Lac91, p. 78], the role of monitoring is to check if the *purpose* of the system is still fulfilled. Correct monitoring requires a model that generates only the desirable behaviors. Such a teleologic monitoring model clearly requires extra knowledge not found in the behavioral model, such as plant safety regulations and cost guidelines that might be violated in some genuine behaviors. In MIMIC such knowledge is supplied through a set of *warning predicates* that return *true* if a state is undesirable in any way. These predicates are used not only to raise immediate alarms but also to forewarn of future undesirable states.

In contrast to identifying undesirable behaviors, we wish to suppress spurious behaviors. A behavior is termed "spurious" if it is physically impossible. Spurious behaviors can arise for several reasons, but two basic categories are: (1) deficiencies in the model, such as neglecting to include a conservation-of-energy constraint, and (2) current limitations in qualitative simulation. Considerable progress has been made in the elimination of spurious behaviors which, in MIMIC, reduces a source of false negatives during fault detection.

3.4.2 Limitations of Alarms

Much of the current practice in process fault detection is based on alarms which are triggered when a measured output exceeds its alarm threshold. Subsequent diagnosis is then based largely on the presence and absence of various alarms. This approach is fundamentally limited because of two forms of information loss. First, diagnosis is based on a very crude quantitative-to-qualitative abstraction of the measured variables; variables are either below normal, normal, or above normal, with no information about how far the alarm threshold has been exceeded and whether the variable is moving back toward normal or further away from it. Second, the sequence of alarms is ignored even though it may contain important clues. As Malkoff explains:

> Current systems ordinarily ignore data about the timing of alarms and other events. They depend on the detection of specific subsets of alarms or, in some cases, the detection of some pre-designated specifically-ordered sequences of alarms. Because the occurrence of fan-in and fan-out, knowledge of only subset membership or sequential order is insufficient for diagnosis. There is need to make use of additional relevant information such as temporal data. But system parameter values and significant events such as alarms are subject to random variability and, therefore, the precise time of occurrence of each new alarm following a malfunction is not fully predictable and is never the same even for identically ordered alarm sequences. [Mal87, p. 99]

MIMIC overcomes both forms of information loss through its use of semiquantitative simulation. First, measurements do not require a quantitative-to-qualitative conversion since predicted ranges are available for direct comparison. If there is a discrepancy, subsequent testing and discrimination of fault hypotheses still uses the unmodified measurement data. Second, MIMIC's diagnosis does not depend upon specific subsets or sequences of alarms. Instead, when a fault occurs and an initial set of discrepancies appear, MIMIC identifies the initial set of suspects through dependency tracing. Fault models are instantiated and simulation begins to reveal, for each model, a branching-time description of behavior. As subsequent manifestations from the same fault appear, tracking will corroborate some

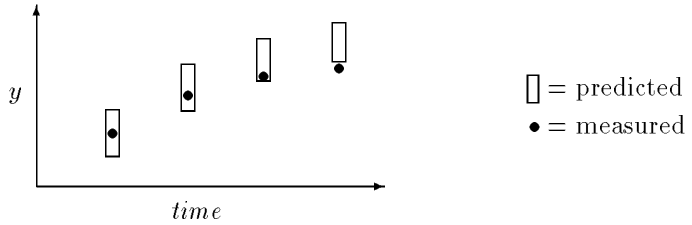


Figure 3.15: The limit test checks each measurement to see if it is within acceptable limits. In MIMIC the limits change dynamically (dynamic thresholds), providing earlier fault detection than with fixed-threshold alarms. This example is shown without noise, but in general, MIMIC assumes that the effect of noise can be adequately modeled by putting intervals around predicted and measured values.

models and refute others. The key point is that simulation predicts all valid orderings of events and tracking corroborates the [few] fault models (and their behaviors) that match the observations over time. Thus, there is no need for rules that attempt to recognize a fault through a specific sequence or subset of alarms.

3.4.3 Discrepancy Detection

Discrepancy-detection is "where measurement meets prediction". The ability to detect disagreement between predictions and measurements is critically important in MIMIC because discrepancies not only detect the existence of a fault in the mechanism but also refute incorrect hypotheses in the tracking set. All additions to and deletions from the tracking set depend on discrepancy detection.

Discrepancy-detection poses two challenges: to extract as much information as possible from predictions and measurements in order to detect anomalies when they occur, and to ensure that a discrepancy is always due to a fault rather than to approximations in the model or "missing predictions". This section describes four methods in the open-ended category of discrepancy-detection methods for continuous deterministic dynamic systems: the *limit* test, *trend* test, *acceleration* test, and *analytic* test. Examples of each test appear in Chapter 4. The tests are described here without noise but, in general, MIMIC assumes that the effects of noise can be adequately modeled by putting intervals around predicted and measured values.

Limit Test The limit test is the simplest and most obvious test of a mechanism's behavior in which a measured value is checked to see if it falls within acceptable limits. Any measurement that goes out of range, as shown in Figure 3.15, is detected by the limit test, whether the divergence is abrupt or gradual.

Figure 3.16: The trend test uses two consecutive measurements (t_1, t_2) to check the rate-of-change of a measured variable. The mean value of the measured rate-of-change must fall within the range predicted for the time interval (t_1 t_2).

Many industrial systems, as noted earlier in section 1.4.4, check easily available signals against fixed limits. MIMIC improves on this approach in two ways. First, the semi-quantitative model automatically computes the limits (bounds) for every variable based on the partial quantitative knowledge in the model rather than on experimentally-set limits. Second, when measurements arrive for time t, MIMIC advances the semi-quantitative simulation to time t, changing all predicted bounds in accord with the model's dynamic behavior. Generating dynamic thresholds in MIMIC eliminates a disagreeable tradeoff that must be made with fixed-threshold alarms — the tradeoff between wide limits, which miss some faults and are slow to detect others, and narrow limits, which give too many false alarms, especially during periods of wide dynamic change such as startup and shutdown.

Trend Test The trend test checks the rate-of-change of a measured state variable. This test is able to detect some deviations in behavior more quickly than the limit test. In industry, the trend test is not used as frequently as the limit test, but when it does appear, it is usually a test against fixed thresholds. As with the limit test, MIMIC improves on this technique by computing the bounds from the partial quantitative knowledge in the model and by changing the bounds as the behavior evolves.

As Figure 3.16 shows, the trend test depends on two consecutive measurements of a variable. The slope between those measurements is the mean value of the rate-of-change. It's important to recognize that the mean value can be reached at *any* instant within the time interval, depending on the dynamics of the mechanism. Thus, the predicted rate-of-change for the time interval must be conservatively stated as the range from the smallest value to the greatest value of the first derivative at the two time points. The dashed box depicts this conservative range.

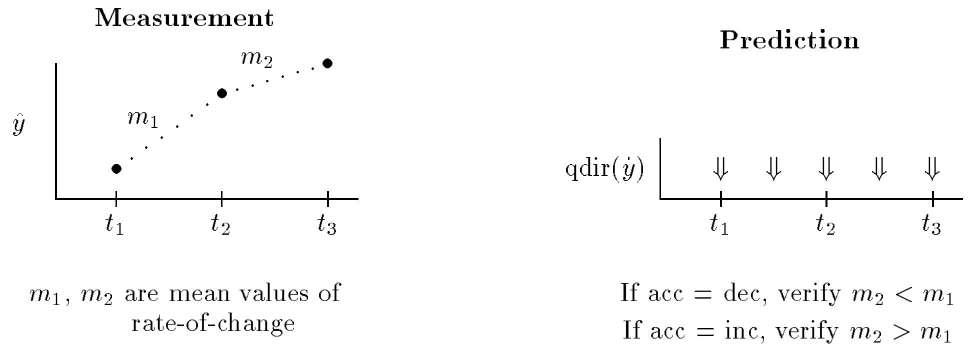


Figure 3.17: The acceleration test uses three consecutive measurements (t_1, t_2, t_3) to check the direction of change of a measured variable's rate-of-change. To apply this test, the predicted qdir of the measured variable's first derivative must remain the same over the open time interval (t_1 t_3).

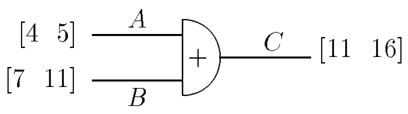
The trend test can only be applied over a time interval in which the sign of the first derivative remains the same. Otherwise, the mean value of the rate-of-change is not properly bounded by the predictions at the two time points. Fortunately, the qualitative behavior generated by QSIM and tracked by MIMIC provides the needed information — as long as the *qdir* of the measured variable remains the same between the two measurements, then it is valid to apply the trend test. This illustrates an important point that we will see again with the acceleration test: *a qualitative description of behavior provides a context that enables a principled interpretation of measurements.* A qualitative simulation guarantees that a predicted *qdir* is unchanging over a qualitative time interval. A conventional numerical simulation cannot provide such a context because it cannot, in general, guarantee that the true behavior is "smooth" between two consecutive steps of simulation.

Unlike the limit test, the trend test is a retrospective test. Given measurements at time t_n, the trend test is really a test on the behavior immediately preceding t_n, specifically the time interval $[t_{n-1}\ t_n]$.

Acceleration Test The acceleration test, as the name suggests, checks the second derivative of a measured state variable. This test is more sensitive than the trend test for certain types of misbehavior, but it uses only a qualitative description of acceleration. Specifically, in the semiquantitative model, only the *sign* of the acceleration of a state variable is known, i.e., the rate-of-change is either increasing, decreasing, or steady.

As Figure 3.17 shows, three consecutive measurements are used in the acceleration test to compute the mean values of two consecutive rates-of-change. The predicted acceleration is the qdir of the first derivative of the measured variable, and the test is ap-

Model & State:



Measurements:

$\hat{B} = [7 \;\; 8]$

$\hat{C} = [15 \;\; 16]$

(Violates $A + B = C$)

Figure 3.18: The analytical test checks for mutual consistency among measurements and known analytical relationships. Even though the two measurements are individually compatible with the current predictions, they are mutually inconsistent with the equation and the assumed value for A.

plied only if the acceleration has remained the same (`inc`, `dec`, or `std`) from t_1 to t_3. If the acceleration is `inc` then verify that $m_2 > m_1$; if the acceleration is `dec` then verify that $m_2 < m_1$; if the acceleration is `std` then verify that $m_2 = m_1$.

Analytical Test The analytical test checks the mutual consistency of a *set* of simultaneous measurements using the known analytical relationships expressed in the model and the assumed values of unmeasured parameters. Very simply, each measurement in a set may be compatible with the model's predictions, but the set as a whole may be inconsistent. Figure 3.18 shows a simple example of this, where measured variables B and C are each compatible with their predicted range, but taken together with the assumed value of A, they violate the model's equation $A + B = C$.

Interestingly, the analytical test is a byproduct of updating the model with each set of measurements. After a set of measurements has passed the limit, trend and acceleration tests, Q2 intersects each measurement with its predicted range, and propagates the resulting range through the model to further narrow the ranges of related variables. If, at any time during this propagation, the range of a variable becomes empty, an analytical discrepancy is declared.

The analytical test subsumes the limit test for limits generated by Q2, since the same predictive machinery is used in both cases. However, limits generated by NSIM are sometimes tighter than those of Q2, so the limit test is still performed.

3.4.4 Tracking

The detection of a discrepancy does not necessarily mean that the current model is incorrect. It may simply mean that the mechanism is operating in a different region of the qualitative behavior than had been assumed. For example, if we are monitoring the temperature in a house during the winter and observe that the temperature is slowly

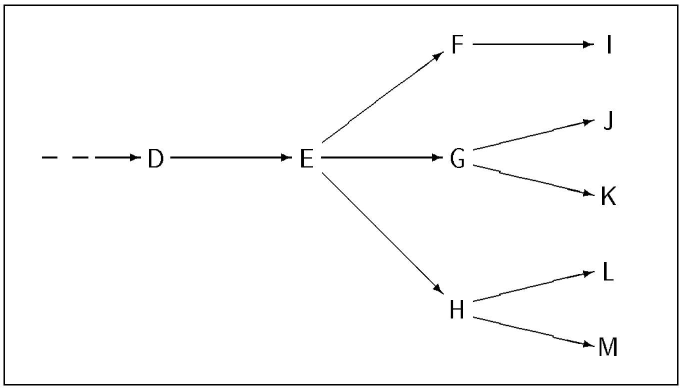


Figure 3.19: Tracking through a behavior graph. As the observed behavior unfolds, MIMIC matches measurements against predictions in the behavior graph, advancing to successor states as needed to track the observed behavior. MIMIC expands only the paths that match the observations.

dropping, it is compatible to assume that the system is operating in the region of qualitative behavior where the furnace is off and the house is slowly cooling. However, when we later observe that the temperature is rising, it does not necessarily mean that a fault has occurred, but rather that the system has "moved forward" in some *predictable* way. In this case, it is predictable that the furnace turns on after the temperature drops below a threshold, and then heats the house causing a rise in temperature. The procedure that MIMIC uses for resolving such discrepancies is called *tracking*.

Tracking is the continuous process of following a path through a behavior tree, as guided by observations. When a discrepancy is detected between observations O and state S in behavior B, tracking examines the immediate successor(s) of S (by traversing the behavior graph B) for compatibility with O. If a compatible successor state is found, then that state replaces S as the currently tracked state. For example, in Figure 3.19, if MIMIC is currently at state E when a discrepancy is detected, tracking will advance to the successor states F, G, and H, testing each one for compatibility with the observations. If only state G is compatible, then it will replace E as the currently tracked state. In effect, discrepancies drive the qualitative simulation forward. In general there may be multiple tracked states for a given model, as shown earlier in Figure 3.2; each state is considered a separate tracking problem.

Tracking may have to move forward by more than one state since measurements may not occur as frequently as events in the simulation. The obvious question arises: how far forward should tracking go in trying to resolve a discrepancy? In MIMIC the search is limited in two ways. First, the tracker checks the lower bound on the time of the state. If that lower bound is greater than the time of the latest measurements, then no further search is attempted in that behavior. However, this method alone is often not enough to limit the search since some states have very conservative lower bounds on time, such as the time of the latest measurements. Thus, a second method is used in which the search is limited to a user-specified number of steps. This number should be based on the longest expected time between measurements and the shortest expected time between successive states in the simulation.

3.4.5 Updating Predictions from Measurements

The last step in the monitoring cycle is to update the current predicted state from the measurements. The principle involved here is that the measurements contain new information that can be used to reduce ambiguity in the state of the model.

The basic computation performed in updating is that for some variable x, the predicted range x_p is intersected with the measured range x_m and the resulting range is propagated through the equations of the model, possibly narrowing the range of other variables and constants (this computation is performed by Q2). It's important to understand why x_p is *intersected* with x_m rather than *replaced* by it. The reason is that we believe in a model only to the extent that its predictions overlap the measurements. For example, if x_p = [3.1 3.7] and x_m = [3.6 3.8], the intersected range [3.6 3.7] is the extent of the agreement, and should be the basis for any future predictions with this model. Of course, when the intersection is empty, there is no agreement, so a discrepancy is declared.

Updating the state provides two benefits. First, and most obviously, it narrows the range of some variables and therefore tightens future predictions from this state. This makes it more likely that the behavior will be refuted if it is incorrect. Second, but no so obvious, updating may narrow the range of some constants (i.e., parameters), thus tightening the model in a different way. For example, consider the simple constraint $C + y = z$ where C is a constant with an assumed range of [7 9], and the values of y and z after intersection with their respective measurements are y = [1.5 1.7] and z = [8.5 10.0]. By propagating ranges through the equivalent constraint $C = z - y$, C's upper bound is lowered from 9 to 8.5. All future predictions involving C will benefit from its more precise value. In short, range propagation in Q2 does not distinguish between variables and constants (nor should it); if the measurements, predictions, and model are consistent with a narrower range for some constant, that range will be updated.

3.4.6 Measurement Issues

- Measurements do not need to be periodic; all that Mimic needs to know with each set of measurements is the time at which they were taken.
- The set of measured variables can change at any time; there is no requirement that the same variables be measured each time. This permits, for example, occasional manual measurements as well as selective sensor focus in large systems.

Identify Suspects:
```
Given a set of discrepancies and the structural model,
identify suspects via dependency tracing.
```
Test:
```
Test each suspect via constraint suspension.
If the test fails, discard the suspect.
```
Refine:
```
Create fault models for each suspect,
discarding any that already exist in the tracking set.
```
Initialize:
```
Attempt to initialize the fault model.
If initialization fails, discard the model.
```
Resimulate:
```
Resimulate the model from successively earlier measurement
times to identify the approximate time of fault.
```

Figure 3.20: The diagnosis algorithm. Diagnosis takes as input a set of discrepancies and produces as output a set of initialized fault models which are added to the tracking set.

- Measurements do not have to appear in chronological order. A measurement taken at an earlier point in time but reported later may be retroactively applied to all behaviors in the tracking set, updating some behaviors and possibly refuting others. One example of such an out-of-order measurement is a blood test, where the sample is taken at time t but the lab results are not available until time $t + 6$.

- Frequent measurements help in detecting faults having compensatory response. If a large amount of time elapses between measurements, some fault manifestations can disappear within that time.

- The more simultaneous measurements, the better. The analytical test is most capable of detecting a discrepancy when it has the maximum number of simultaneous measurements.

3.5 Diagnosis

Diagnosis is the process of isolating a fault to a specific misbehavior of a specific component or parameter. The diagnosis algorithm, as shown in Figure 3.20, takes as input a discrepant model-behavior and generates a set of initialized fault models which are added to the tracking set. These models are then tracked against future measurements to refute incorrect models and corroborate valid models.

Each model-behavior in the tracking set is a hypothesis about the state of the mechanism and, as such, has two attributes: probability and similarity. These attributes are described more fully in section 3.6.3, but for now, simple definitions will suffice. A model's *probability* estimates the likelihood of the model based on the *a priori* probabilities of its individual component operating modes. A behavior's *similarity* is a measure of how well its current predictions match current observations.

3.5.1 Hypothesis Generation

When to hypothesize? In conventional model-based diagnosis where only a single fault-free model is used, there is no question about when to generate hypotheses — the answer is whenever there is a discrepancy between predictions and observations. However, in MIMIC where there may be several different models in the tracking set at any given time, when is it appropriate to generate new hypotheses? The two extreme positions on this question are "whenever any model has a discrepancy" and "when the last model is refuted". We examine these two positions below and show that a better approach is to base the decision on the current state of the diagnosis.

If hypotheses are generated whenever any model has a discrepancy, this can lead to the phenomenon of "chattering hypotheses". Consider the simple case of a tank whose drain-rate is either `normal`, `low`, or `very-low`. Let's say that the tank drain is initially `normal` when it becomes partially obstructed, corresponding to a rate of `low`. When the discrepancy with the normal model is first detected, two fault models are built: one for `low` and one for `very-low`. Since the fault corresponds to a drain-rate of `low`, the model for `low` will be corroborated by future readings, and the model for `very-low` will eventually exhibit a discrepancy. If, at that time, new hypotheses are generated based on a discrepancy with the model for `very-low`, then two hypotheses will be proposed: `normal` and `low`. Since the model for `low` already exists in the tracking set, it will not be built anew, but the model for `normal` *will* be built and added to the tracking set. The normal model will eventually exhibit a discrepancy, starting the cycle all over again. The net effect is that while the correct hypothesis (`low`) is tracked, there will be an endless chattering between `normal` and `very-low` as alternate hypotheses. This is clearly undesirable, and the problem gets worse with large numbers of alternate hypotheses.

At the other extreme, hypothesis generation may be suppressed until the last model in the tracking set is refuted. In other words, as long as there is at least one other model in the tracking set, no new hypotheses will be generated when a model is refuted. This approach eliminates the problem of chattering hypotheses, but it may discard valuable clues. For example, suppose that there are two models in the tracking set — one representing a high-probability fault with high similarity (which happens to be the correct hypothesis at the moment) and one representing a low-probability fault with low similarity. If discrepancies are suddenly detected with both models (because, say, a new fault has occurred) and they are refuted in the order listed above, then hypothesis generation will be based on the incorrect model. The best clues — the discrepancies from the strongest model — will have been ignored, and the single-change hypotheses resulting from discrepancies with the weak model will not contain the correct combination of faults.

A better approach is to include the current state of the diagnosis in the decision of when to generate hypotheses. Specifically, if after performing discrepancy-detection on all models in the tracking set, there is at least one "strong" model, then don't generate hypotheses. Otherwise, generate hypotheses using discrepancies from the strongest of the refuted hypotheses. In the current implementation, a hypothesis is "strong" if it exceeds user-specified minimum thresholds for similarity and probability. This may be improved in the future to make the thresholds change dynamically based on the contents of the tracking set.

Identifying Suspects via Dependency Tracing Given a set of discrepancies found during monitoring, the task of hypothesis generation is to identify all the components and parameters whose malfunction could have caused the discrepancies. MIMIC accomplishes this using the standard technique of dependency tracing, as shown in Figure 3.21.

Dependency tracing is a simple graph-traversal procedure which starts from the site of a discrepancy in the structural model and traces upstream from there, identifying all the components and parameters whose malfunction could have contributed to the discrepancy. The structural model is a directed graph where directions indicate direction of effect, so the algorithm only traces connections in an upstream direction. This means that a component can be entered through an output or 2-way terminal and exited through an input or 2-way terminal. The algorithm must keep track of where it has been to avoid an endless loop when tracing feedback loops.

When there is a set of simultaneous discrepancies, the sets of suspects resulting from individual discrepancies are intersected on the assumption that there is a single fault or repair responsible for those discrepancies. The final set of suspects becomes input to the candidate generation process.

Candidate Generation Given a set of suspects, the next task is to generate modifications of the current model based on the possible fault modes of the suspects. Every component and parameter has an associated set of fault modes. For a parameter, the modes cover all possible value ranges in the parameter's quantity space. For a component, the modes cover all known faults of that type of component by determining which constraints are active (as described earlier in section 3.2.3). Since a component may fail in a novel way, the modeler may include a fault mode that corresponds to *no* constraints (constraint-suspension), thus covering all possible fault behaviors.

A mode is basically a triple $\{F, T, C\}$. F (fault-p) is a truth value indicating

```
dependency-trace (discrepancies)
    z ← {all possible suspects}
    ∀d ∈ discrepancies:
        y ← find-suspects(d)
        z ← y ∩ z
    return z

find-suspects (x)
    If x is traced, return nil
    else mark x as traced.
    If x is a parameter, return {x}.
    If x is a connection or terminal, and the quantity
        at x is measured and not discrepant, return nil.
    y ← trace-upstream(x)
    If x is a component, return x ∪ y
    else return y

trace-upstream (x)
    If x is a terminal
        If x is an output or 2-way terminal,
            return find-suspects( component(x) )
        else return nil
    If x is a component
        y ← nil
        ∀t ∈ terminals(x)
            If t is an input or 2-way terminal
                y ← y∪ find-suspects(t)
        return y
    If x is a connection
        y ← nil
        ∀c ∈ connected-to(x)
            y ← y∪ find-suspects(c)
        return y
```

Figure 3.21: Dependency tracing algorithm. Top-level function `dependency-trace` takes as input a set of discrepancies and returns a set of suspected components and parameters. The lower-level functions `find-suspects` and `trace-upstream` are mutually recursive; each takes an object as input (a terminal, component, connection, or parameter) and returns a set of suspects. The algorithm keeps track of where it has been to avoid endless loops in feedback circuits, and takes advantage of non-discrepant measurements to halt further tracing on some paths.

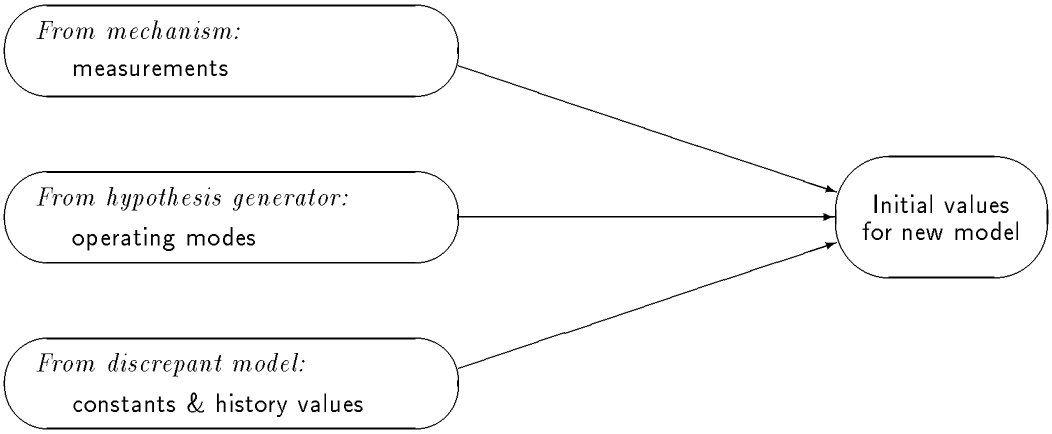


Figure 3.22: Three sources of information are used to initialize a new model: the mechanism provides measurements, the hypothesis generator provides hypothesized operating modes, and the discrepant model provides quantities that cannot change instantaneously—constants and history values. Measurements take precedence over modes, constants, and history values.

whether this mode is a fault mode or a normal operating mode. MIMIC uses F only during the display of hypotheses to distinguish faults from non-faults; it does not use F during diagnosis since it hypothesizes repairs as well as faults. T (type) specifies whether this mode occurs abruptly or gradually. If it occurs abruptly, the mode is always hypothesized; if it occurs gradually, the the mode is hypothesized only if it is adjacent to the current mode of the suspect. C (constraints) specifies the constraints that are active in this mode. In the case of a parameter, C is always nil.

3.5.2 Hypothesis Testing

When a modified model has been hypothesized, the next step is to test the hypothesis by attempting to initialize the model. As Figure 3.22 shows, initial values are taken from three sources of information: the mechanism provides measurements, the hypothesis generator provides hypothesized operating modes, and the discrepant model provides constants and history values. Measurements take precedence over modes, constants, and history values because they come from observations rather than assumptions or predictions.

Since the occurrence of a fault may cause discontinuous changes in behavior, MIMIC must be careful about what values from the state of the old model it uses to initialize the new model. Initial values are taken in decreasing order of preference from the following list:

1. *Measurements.* A symbolic measurement, such as whether a switch is on or off, is used

directly as an initial value. A quantitative measurement, however, is first converted to a qualitative value in order to initialize the qualitative part of the model. The qmag is constructed simply as the interval between the two landmarks that bound the range of the quantitative measurement.

2. *Modes.* Mode variables are always set by MIMIC to specify a hypothesis. Hence, every mode variable is guaranteed to be initialized.

3. *Constants.* A constant may be a "natural constant" used in the model, such as π, or a system parameter whose value is not considered open to suspicion (else it would have been a mode variable). Not only is the qmag initialized but also the qdir since the qdir of a constant is always `std`.

4. *History variables.* A history variable, sometimes called a state variable, is either an integrated quantity or functionally related to an integrated quantity, and therefore cannot change magnitude instantaneously. Therefore, the qmag is inherited from the old model, but the qdir is not since a qdir *can* change instantaneously.

All remaining variables in the new model are left uninitialized. Their values will be determined through propagation and constraint satisfaction.

In all cases, the qdir of an initial value is determined by the *new* model, never by the old model. For example, a variable y might be a time-varying dependent variable in the old model but change to a constant in the new model, as in a "stuck-at" fault. The simple rule of qdir initialization is this: if the variable being initialized is a constant in the new model, then its qdir is set to `std`, otherwise it is set to `nil`. There is no special exemption for measured variables or history variables. Even though the trend of measured values for a variable z has clearly been increasing over the last two or more measurements, there is no guarantee that z is still increasing at the instant of the most recent measurement. Hence, the qdir of a measured value must be set to `nil`. Similarly, a history value cannot inherit its qdir from the state of the old model because a fault can cause an abrupt change in a variable's qdir. Hence, the qdir of a history value must be set to `nil`.

After all initial values have been determined, QSIM/Q2 is invoked to form an initial state through propagation and constraint satisfaction. Two outcomes are possible—initialization either succeeds or fails. If the model is overconstrained by the initial values, then no consistent initial state will be found. In this case, the hypothesis represented by the model has failed the test, and is discarded. If initialization succeeds, the initial state(s) are added to the tracking set. It's important to note that there may be more than one initial state if the model is underconstrained by the initial values. This simply means that the hypothesis represented by the model has more than one possible behavior, given incomplete knowledge of its initial state. In either case, tracking of the state(s) against future measurements will determine if the hypothesis survives.

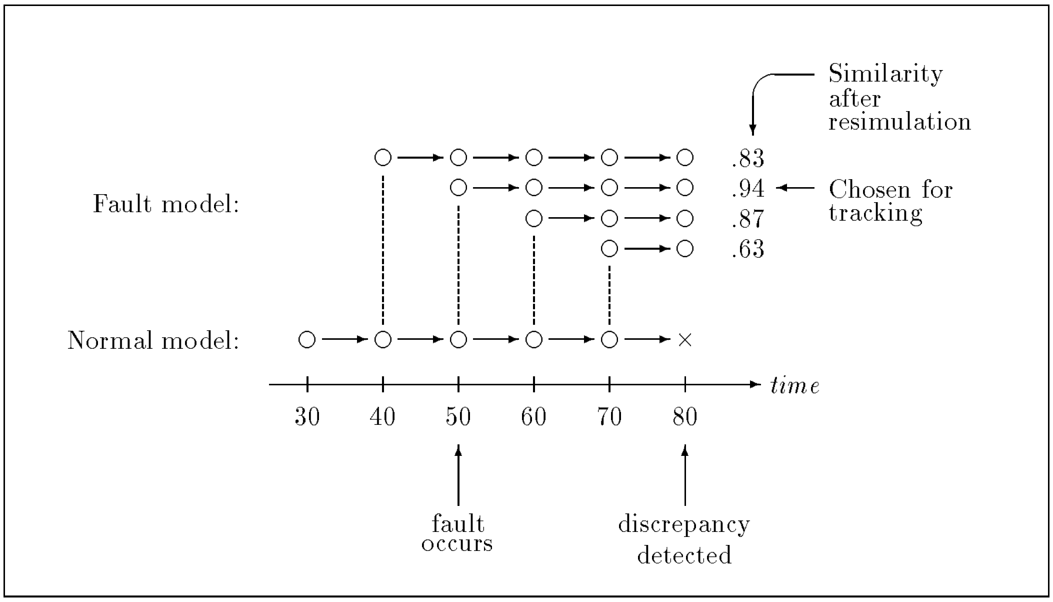


Figure 3.23: Resimulation of a fault model. Tracking of the normal model proceeds until a discrepancy is detected. When a fault model is hypothesized, it is initialized from successively earlier measurement states and simulated up to the time of discrepancy until the ending similarity stops improving. The initialization yielding the highest similarity to the measurements provides the best estimate of time-of-fault and the best corrected state for tracking.

3.5.3 Resimulation

Operative diagnosis differs from maintenance diagnosis in that the effects of a fault are still propagating through the mechanism. Thus, it's important to know *when* a fault occurred in order to predict its current and future effects. There may be an arbitrary amount of time between the occurrence of a fault and its detection through a discrepancy. Thus, by the time the fault is detected, the predicted values for an unmeasured state variable may be significantly different from its true value. Since unmeasured state variables are initialized by inheriting their last predicted value before the discrepancy, the fault model's initial state may be significantly different than the true state.

MIMIC addresses this problem through a procedure called *resimulation*, depicted in Figure 3.23. The basic idea is simple: initialize the new model from successively earlier measurement states, simulating each one up to the present time until the degree-of-match with the last set of measurements stops improving. The starting measurement state yielding the highest degree-of-match represents the best estimate of the time of fault, and the new model's resimulated state represents the best estimate of the mechanism's true values. In short, resimulation is a hill-climbing algorithm that seeks the best state for the new model.

The algorithm is summarized below:

1. Let t_n be the time at which the discrepancy occurred with model M_x. Therefore, the measurement state at t_{n-1} is the last consistent state of M_x.
2. Let $i \leftarrow (n-1)$, $e \leftarrow 0$.
3. Given a new model M_y to initialize and test, initialize M_y from the measurement state of M_x at t_i, simulate it up to time t_n, updating its state along the way with each intervening set of measurements, and compute its degree-of-match d with the measurements at time t_n.
4. If $d \leq e$, done. Otherwise, set $e \leftarrow d$ and $i \leftarrow (i-1)$.
5. If $i < 0$, done. Otherwise, go to step 3.

Resimulation is important not only to correct the state of the model but also to inform the operator of how long the hypothesized fault has existed. To an operator, there's a big difference between knowing that a leak has just occurred versus learning that the leak started forty minutes ago. The operator can use the estimated time-of-fault to estimate the amount of damage and/or urgency of a fix.

Resimulation requires Mimic to retain a lot of data. Let's examine an extreme example to see if the storage requirements are reasonable. Assume that 100 readings are taken every second and that each reading requires 8 bytes of memory, so storage is consumed at the rate of 800 bytes/second. Also, assume that the model has 100 state variables (it may have many other non-state variables, but they don't need to be saved), and since predictions are made for each measurement instant, another 800 bytes/second is consumed. If measurements are to be retained for the last hour, they can be saved in less than 6 MB of storage. In short, resimulation does not demand an unreasonable amount of storage on modern computers.

3.5.4 Hypothesis Discrimination

Hypothesis discrimination is the task of obtaining additional information in order to discriminate among multiple hypotheses. In a continuous monitoring system such as Mimic, new information is arriving all the time, so the task is accomplished in a natural way: each new set of measurements tests every hypothesis in the tracking set, and hypotheses that fail the test are discarded.

Two other techniques are available to help in hypothesis discrimination, both at the discretion of the operator. The operator may manually measure an unsensed variable and supply that measurement to Mimic, or the operator may perturb one or more measured system inputs and let the resulting perturbations in sensor measurements serve as diagnostic clues. MIMIC does not currently offer advice about what variables to measure or what inputs to perturb; the minimum-entropy method for measurement selection employed in GDE [dKW87] and SHERLOCK [dKW89] offer guidance in this area of future work.

3.5.5 Multiple-Fault Diagnosis

MIMIC makes a simplifying assumption that faults occur one-at-a-time with respect to the sampling rate. For a system of n components, this assumption reduces the size of the hypothesis space from $O(2^n)$ to $O(n)$. MIMIC is still able to form multiple-fault hypotheses, but does so one fault at a time. In each cycle of the main loop, MIMIC may hypothesize new faults or repairs, so models of varying numbers of faults may be found in the tracking set.

3.6 Advising

The purpose of the advising task is to interpret the state of diagnosis for the operator. This task of interpretation has three parts: (1) warning of current undesirable behavior, (2) forewarning of potential undesirable behavior in the near future, and (3) presentation of a ranked set of fault hypotheses.

3.6.1 Warning Predicates

In conventional alarm-based monitoring, an alarm draws the operator's attention to an abnormal value, but it doesn't say whether the alarm was caused by a fault or is simply undesirable behavior. This may seem like a subtle distinction, but the fact is that there is no direct relation between faults and unacceptable behavior. A mechanism containing a fault, such as a tank with a minor obstruction in its drain, may still operate within an acceptable range. Conversely, a fault-free mechanism may produce undesirable behavior, such as a bathtub that overflows because the inflow rate is excessive, even though nothing is wrong with the bathtub itself. In short, MIMIC cannot draw conclusions about the acceptability of a behavior based on fault hypotheses; extra domain knowledge is required.

MIMIC distinguishes between desirable and undesirable behavior through a set of warning predicates, as noted earlier in Figure 3.14. Warning predicates can be applied to a set of measurements to identify current undesirable behavior of the mechanism, and they can also be applied to a predicted state in order to forewarn of possible future undesirable behavior. Unlike alarms, which are based on measured quantities, warning predicates can also be based on unmeasured quantities in a model's state. Thus, warnings can be based directly on the quantities of interest, regardless of their measurability.

3.6.2 Forewarning

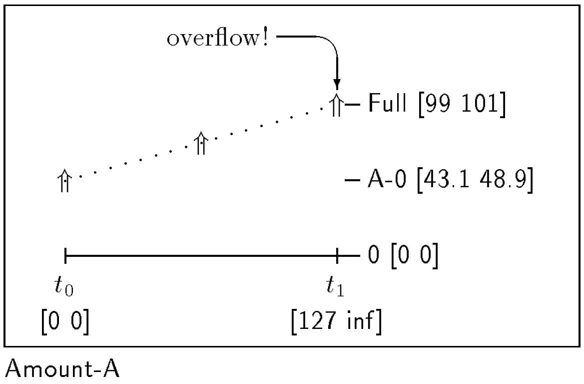


Figure 3.24: Forewarning of future undesirable behavior. By simulating ahead in *qualitative* time, possible future undesirable states can be identified along with an estimated time of occurrence. In this example, overflow of a tank is possible as early as $t = 127$ or as late as infinity (meaning that it will not necessarily occur), based on current measurements.

MIMIC is capable of forewarning the operator of near-future undesirable states. This is done efficiently by simulating ahead in qualitative time and testing the resulting states with the warning predicates. Figure 3.24 shows a simple example where the possibility of an overflow is predicted. Importantly, semiquantitative simulation computes bounds on the time at which overflow is reached, so the operator can know how soon the problem may occur.

Two important properties of the forewarning algorithm are that it is sound and efficient:

- The forewarning algorithm can guarantee that *all* near-term future states of a given hypothesis are predicted because qualitative simulation is used.
- The forewarning algorithm is efficient because the lookahead is accomplished with *qualitative* simulation. In contrast, simulating ahead in *quantitative* time would require hundreds or thousands of simulation steps.

3.6.3 Ranking of Hypotheses

At any given time the tracking set may contain multiple hypotheses. Each hypothesis is consistent with the most recent readings, of course, otherwise it would not be in the tracking set. But there is additional information about each hypothesis that can help the operator focus attention on a subset of the hypotheses. In the following list we present four methods of ranking hypotheses:

Age: A hypothesis "lives" in the tracking set from the moment it is hypothesized until it exhibits a discrepancy. Since new hypotheses may be generated on any cycle of the main loop, the tracking set may contain hypotheses of varying age. The oldest hypothesis has the most corroborating evidence since it has survived the greatest number of tests (each set of observations is a test). Thus, the oldest hypothesis may be favored over all others, but it must be remembered that this ranking strategy is biased against new evidence.

Probability: Given the *a priori* probability of each fault in a model and the assumption that faults are independent, MIMIC computes the probability of the model as a whole. In situations where there are large differences in fault probabilities, this metric usefully ranks the hypotheses by likelihood. This is particularly useful when the hypotheses are ranked nearly equal by the other metrics.

Similarity: The similarity value computed by MIMIC is a measure of how well the predictions of a model match the observations. The calculation produces a real number between 0 and 1, where 1 occurs when the midpoint of the observed range equals the midpoint of the predicted range, and 0 occurs when there is no overlap between the two ranges. For multiple variables, overall similarity is the minimum of the individual similarities.

It's important to remember that as long as similarity is greater than zero, the model (i.e., the hypothesis) cannot be discarded. For this reason, similarity itself is not a good metric for ranking hypotheses. However, a negative rate of change in similarity means that predictions are diverging away from observations, often foretelling the eventual refutation of the hypothesis. Thus, ranking by similarity's rate of change *is* informative.

Risk: By simulating ahead in qualitative time, MIMIC is able to forewarn of undesirable near-future states. The undesirable states are identified by applying the warning predicates, and are ranked by undesirability using domain knowledge.

The choice of which ranking metric (or combination of metrics) to use is clearly domain-dependent. For example, where safety is the overriding concern, the hypotheses should be ranked first by risk and second by probability.

3.6.4 Defects vs. Disturbances

A fault may be either a defect or a disturbance. A *defect* is a fault whose cause is internal to the mechanism, such as a component that is broken or out of calibration, or a connection that is severed or blocked. Diagnosis of a defect calls for repair of the mechanism or, if that is not immediately feasible, for reconfiguration or compensation. A *disturbance* is a fault whose cause is external to the mechanism, such as an input that is abnormally high. Diagnosis of a disturbance calls for correction of the environmental inputs rather than repairs to the mechanism. Thus, in order for the operator to know what response is appropriate. the advising task must identify each hypothesized fault as a defect or a disturbance. This is accomplished by simply annotating which parameters represent inputs.

3.7 Special Fault Handling

3.7.1 Intermittent Faults

Since MIMIC performs continuous monitoring and diagnosis, it has the potential to identify intermittent faults. By keeping simple statistics on each possible fault, MIMIC accumulates evidence of intermittency. Specifically, two statistics are kept: the number of times that the fault has been hypothesized and refuted, and the average length of time the fault hypothesis survived. Faults that score high in either measure may be intermittent.

3.7.2 Consequential Faults

Sometimes a failure in one component predictably causes a failure in another component. For example, if two electrical heating elements are in series and one of them fails with a short-circuit across its terminals, the other element will draw much more current and dissipate much more power than it was designed for, and will fail in a short time. This is an example of a *consequential fault* where the effects of one fault propagate to other components, stressing them beyond their design limit.

Consequential faults can be diagnosed in MIMIC to the extent that the individual component models can predict their own failure based on abnormal inputs. For example, a heating element might predict its own failure as a non-linear function of power dissipation and time, using envelope functions to represent upper and lower bounds on lifetime. When the lifetime threshold is reach, the component would change its own mode from `normal` to `burned-out`. Thus, a new fault would be hypothesized not as a result of discrepancy detection but rather through prediction.

3.8 Controlling Complexity

There are two main sources of combinatoric explosion in MIMIC: prediction of an explosive number of behaviors during simulation and generation of an exponential number of hypotheses during diagnosis. We examine each problem below and explain the practical methods used to control it.

Semiquantitative simulation can generate a large tree of behaviors because the simulation branches on every qualitative distinction. This problem of “intractable branching” is partly a consequence of being able to simulate with incomplete knowledge and partly a consequence of the relatively early stage of development of semiquantitative mathematics. Since this problem doesn't exist in conventional numerical simulation, it is well to recall why we are using semiquantitative simulation in the first place. Any real mechanism operating within normal limits can exhibit an infinite number of infinitesimally different behaviors. By using semiquantitative simulation we can cover an infinite number of real behaviors with a single semiquantitative behavior. While mean-and-variance simulation can cover minor variations in a real behavior, it does not guarantee to predict all possible behaviors of the mechanism, such as whether or not a rocket achieves escape velocity.

While some branching is unavoidable in semiquantitative simulation because of incomplete knowledge, there are practical steps that the modeler can and should take to reduce the size of the behavior tree.

- Many variables include `inf` and/or `minf` in their quantity spaces, but the landmark itself is usually not reachable in a realistic system. By appropriately specifying unreachable values in the QDE, some impossible behaviors will be eliminated.

- The "qdir" (qualitative direction of change) of a derivative variable is often unconstrained and can introduce unwanted distinctions in behavior. These distinctions can often be eliminated through the use of higher-order derivative constraints or suppressed with `ignore-qdirs`.

- Partial quantitative knowledge can substantially improve the precision of a model and thereby eliminate behaviors inconsistent with that knowledge. The knowledge is expressed in the form of initial ranges for landmark values and envelope functions for monotonic function constraints. The more precise that this knowledge is, the more effective it is in reducing the number of behaviors.

- QSIM offers several global filters to eliminate spurious behaviors, such as the analytic function filter, the non-intersection filter, and the energy filter. All appropriate global filters should be used during simulation.

In addition to what the modeler can do, two aspects of MIMIC's design help control the proliferation of behaviors:

- MIMIC simulates incrementally, advancing only as far as needed to check current measurements and forewarn of imminent undesirable states. As measurements refute some behaviors, they are pruned, thereby reducing the number of paths that MIMIC will expand and explore in the behavior tree.

- Each time that measurements are unified with predictions, the state of the model is made more precise, potentially eliminating some branches in the behavior tree.

Hypothesis generation and testing is inherently exponential in the number of fault hypotheses to consider. Given a mechanism of n components, each of which has an average of m operating modes (of which $m-1$ modes are considered fault modes), there are m^n possible fault combinations. MIMIC reduces this complexity in the following ways:

- The single-fault-at-a-time assumption reduces the number of hypotheses to $(m-1)n$.
- Constraint-suspension is used to test a suspect before instantiating and testing its $m-1$ fault modes. This provides early elimination of some suspects.
- The distinction between abrupt faults and gradual faults reduces the number of fault modes for a gradual-fault suspect from $m-1$ to 1 or 2.
- The four discrepancy-detection methods described earlier provide strong tests on new and existing hypotheses. Of course, these methods depend on good sensor placement and adequate model precision.

Chapter 4

Experimental Results

The success of a paradigm—whether Aristotle's analysis of motion, Ptolemy's computations of planetary position, Lavoisier's application of the balance, or Maxwell's mathematization of the electromagnetic field—is at the start largely a promise of success discoverable in selected and still incomplete examples.

The Structure of Scientific Revolutions
Thomas S. Kuhn [Kuh70]

The purpose of this chapter is to illustrate the operation of MIMIC through example. Using simple fluid-flow mechanisms, we show the flow of control and data during monitoring and diagnosis, and strive to "make visible" the computation of several key algorithms in MIMIC.

The preceding chapter has described the many elements of MIMIC's design — semiquantitative simulation, state-insertion for measurements, discrepancy-detection, state updating, tracking, dependency-tracing, model modification, initialization, resimulation, hypothesis ranking, forewarning — and this chapter illustrates each one in operation. Particular attention is given to discrepancy-detection because of its central role in fault detection, hypothesis discrimination, and control of complexity.

4.1 Gravity-Flow Tank

We begin with the simplest possible dynamic system in order to illustrate MIMIC's operation without the distraction of domain complexity. The gravity-flow tank, as shown in Figure 4.1, is simply a tank that begins full and drains toward empty. The only measured variable is the amount of water in the tank. The purpose in monitoring the draining of the tank is to detect a possible obstruction in the drain.

Figure 4.2 shows the semiquantitative model. To keep this example utterly simple, the outflow rate has been modeled as being linearly proportional to the amount of water in the tank. In a more realistic model, outflow rate would be proportional to the square-root of the drain pressure (as shown earlier in chapter 3).

Our monitoring scenario begins at $t = 0$ with a full tank, when draining commences. Measurement of the amount of water in the tank is taken every 6 seconds (every

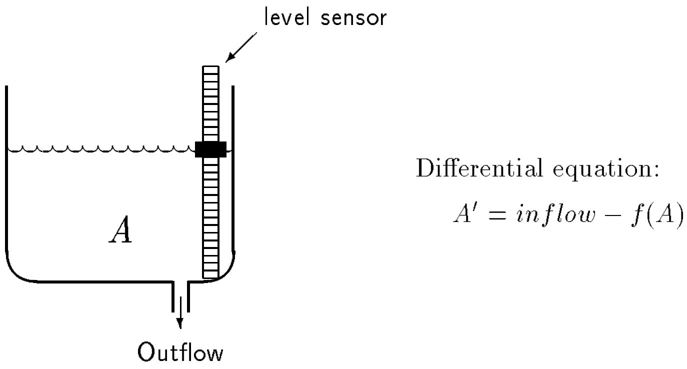


Figure 4.1: Gravity-flow tank. The amount of water in the tank is monitored as the tank drains due to gravitational flow.

```
(define-QDE GRAVITY-FLOW-TANK
  (quantity-spaces
    (Amount       (0 Full))
    (Outflow      (0 Omax))
    (Netflow      (minf 0 inf))
    (Drain        (0 hi-blockage lo-blockage normal)))

  (constraints
    ((MULT  Amount Drain Outflow)  (Full normal Omax))
    ((MINUS Outflow Netflow))
    ((D/DT  Amount Netflow)))

  (history      Amount)
  (independent  Drain)
  (unreachable-values (Netflow minf inf))

  (initial-ranges ((Time   T0)           (0    0))
                  ((Amount Full)         (99   101))
                  ((Drain  normal)       (0.8  1.0))
                  ((Drain  lo-blockage)  (0.4  0.8))
                  ((Drain  hi-blockage)  (0.0  0.4))))
```

Figure 4.2: Semiquantitative model of the gravity-flow tank. Incomplete quantitative knowledge is expressed in the form of ranges for landmark values, in `initial-ranges`. The `Drain` values `lo-blockage` and `hi-blockage` are used in fault models to simulate a clogged drain.

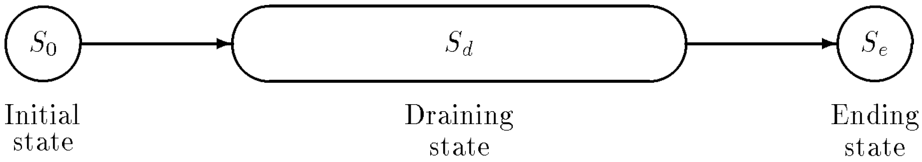


Figure 4.3: Behavior tree for the gravity-flow tank. The initial state S_0 and the ending state S_e represent instants in time, whereas the draining state S_d represents an interval of time. During monitoring, measurement states are inserted into this time interval.

0.1 minute). Immediately following the measurement at $t = 1.0$, the drain becomes partially obstructed, reducing its flow rate by a third. In the following narrative, we examine three cycles of MIMIC's main loop: one before the obstruction occurs, one when the anomaly is detected, and one when an incorrect hypothesis is refuted. The ninth cycle, at $t = 0.9$, illustrates four basic procedures in monitoring: state-insertion for measurements, semiquantitative simulation, discrepancy-detection, and state updating. The eleventh cycle, at $t = 1.1$, which actually detects a fault, illustrates five more procedures: tracking, dependency-tracing, model modification, initialization, and resimulation. The entire monitoring history from which this narrative is derived is provided in Appendix A.

4.1.1 Cycle 9: t = 0.9

State-insertion for measurements The arrival of new measurements necessitates the creation of a new state and its insertion into the behavior graph. Let's refer to the previous measurement state as $S_{0.8}$ and the new measurement state as $S_{0.9}$. Further, let's refer to the three states in the attainable envisionment as S_0 (initial state), S_d (draining state), and S_e (empty state), as shown in Figure 4.3. To determine where to insert $S_{0.9}$ in the behavior tree, MIMIC inspects the time range of the next point-state following $S_{0.8}$, which in this case is S_e with a time range of [inf inf]. Hence, it is temporally compatible to insert $S_{0.9}$ in the time interval between $S_{0.8}$ and S_e.

Semiquantitative simulation Given the previous state of the model at $t = 0.8$ and the arrival of new measurements for $t = 0.9$, the purpose of semiquantitative simulation is to predict the values of all variables for $t = 0.9$. At the end of the previous cycle at $t = 0.8$, amount was the range [47.21 50.13]. For $t = 0.9$ Q2 predicts [42.20 46.75] and NSIM predicts [42.72 46.28]. These two ranges are intersected to obtain the tightest

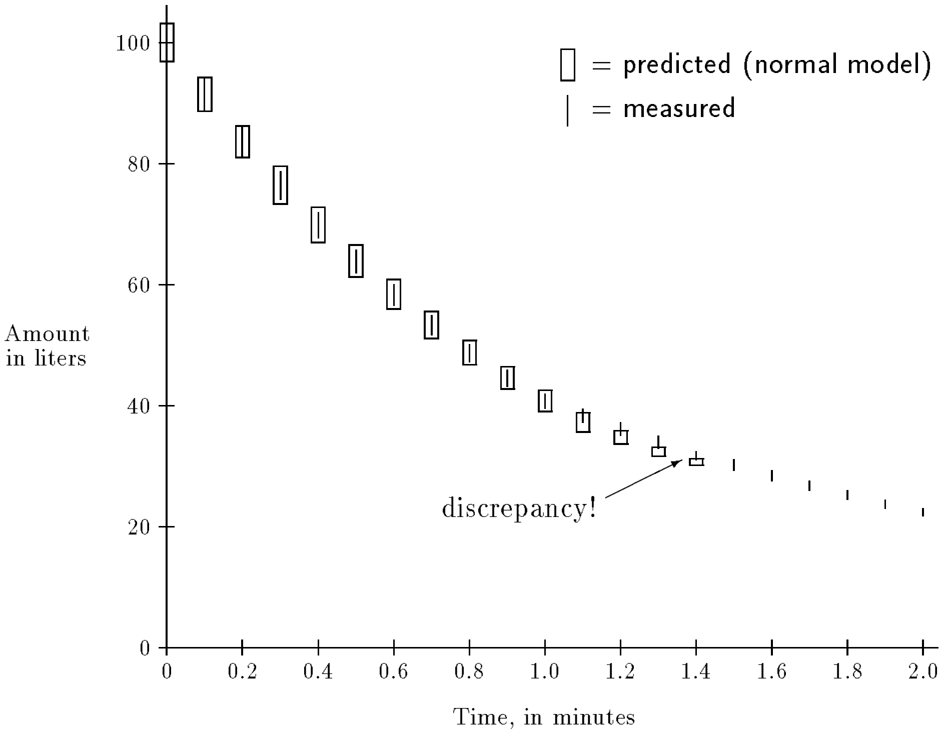


Figure 4.4: Gravity-flow tank: predictions from the normal model, plus measurements. After the drain becomes partially obstructed at t = 1.001, measurements start to diverge from predictions until the limit test fails at t = 1.4. However, as the next figure shows, the trend test detects a discrepancy much sooner.

prediction. In this case, NSIM provides the tightest prediction for both upper and lower bounds.

Discrepancy-detection The new measurements at $t = 0.9$ are subjected to four discrepancy-detection tests. As Figure 4.4 shows, the limit test passes since there is overlap between the predicted range of [42.72 46.28] and the measured range of [43.15 45.81]. Likewise, as Figure 4.5 shows, the trend test passes since there is overlap between the predicted range of [-50.13 -34.17] and the measured range of [-43.16 -40.64]. Also, the acceleration test passes since the predicted acceleration agrees with the measured acceleration.

State updating The last discrepancy test—the analytical test—is a byproduct of updating state $S_{0.9}$ with the measurements from $t = 0.9$. For each measured quantity, the measured range is intersected with the corresponding predicted range, and the result is propagated through the equations of the model, possibly updating other values. In this example, the only measured quantity is `Amount`. The state value for `Amount` in

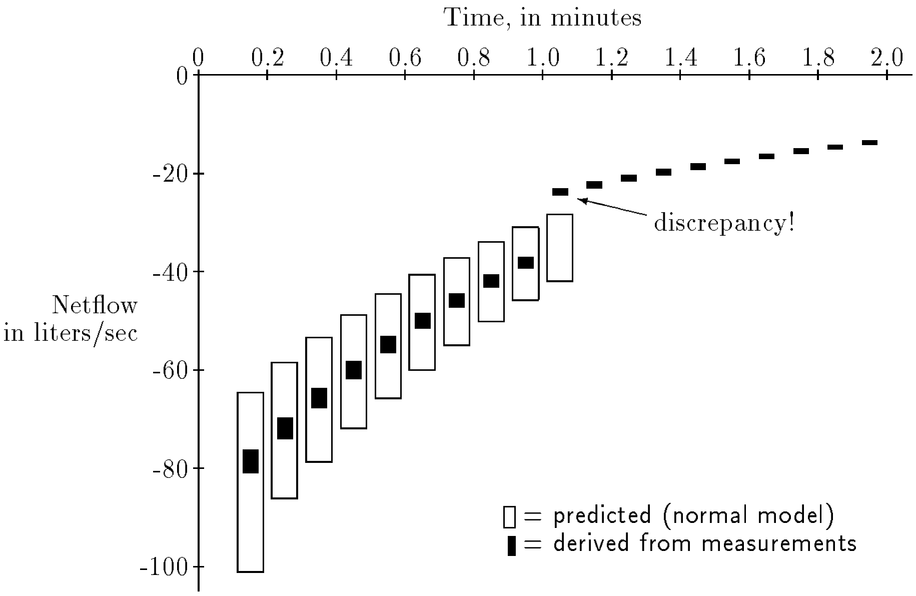


Figure 4.5: Gravity-flow tank: measured and predicted trends. After the drain becomes partially obstructed at $t = 1.001$, the trend test detects a discrepancy on the next measurement, much sooner than the limit test. Notice that the trend predictions cover the *interval* between two successive measurements, and the measured trend is really the mean-value range of the rate-of-change, as derived from the measurements.

$S_{0.9}$ is replaced by the intersection of measurement and prediction (specifically, [43.15 45.81]). The resulting propagation does not tighten any other values.

4.1.2 Cycle 11: t = 1.1

Discrepancy-detection Immediately after the tenth monitoring cycle at $t = 1.0$, the tank drain became partially obstructed, reducing its flow rate by one-third. The resulting perturbation in behavior passes the limit test at $t = 1.1$ but fails the trend test, as shown in Figure 4.5. As Figure 4.4 shows, the limit test would have eventually detected a discrepancy at $t = 1.4$, but the trend test is more sensitive to this type of fault and detects it sooner.

Tracking MIMIC's first reaction to a discrepancy is to attempt to resolve the discrepancy by moving forward in the qualitative behavior tree. In this example, the only unexplored region of behavior (besides the time interval preceding S_e, which was being tracked) is the final state S_e. Accordingly, MIMIC tries to create a measurement state $S_{1.1}$ on top of S_e but fails because the time range of $S_{1.1}$ ([1.1 1.1]) has no overlap with the time range of S_e ([inf inf]). Hence, the discrepancy is not resolved and is thus assumed to be due to a fault.

Dependency-tracing The dependency tracing algorithm takes as input a set of one or more simultaneous discrepancies and produces as output a set of suspects. In this case the only discrepancy is for `Amount`. Tracing upstream from `Amount` in the structural model identifies three suspects: the tank, the drain, and the amount sensor. In the spirit of keeping this example utterly simple, the tank and amount sensor are assumed to be fault-free, so the only real suspect is the drain.

Model modification Given the drain as a suspect, the task of model modification is to create single-change variations of the discrepant model. The `drain` parameter has three possible values: `normal`, `lo-blockage`, and `hi-blockage`. Since the value of `drain` in the discrepant model is `normal`, two new models are created, one for `drain` = `lo-blockage` and another for `drain` = `hi-blockage`.

Initialization & Resimulation The purpose of initialization is to attempt to initialize each new model from the last consistent state of the now-discrepant model using observations, modes, constants, and history variables, as described in section 3.5.2. Resimulation reattempts initialization at successively earlier times in a hill-climbing search for the time-of-fault. In this case, the model for `drain` = `hi-blockage` initializes with highest similarity of 0.492 from time 1.0; the model for `drain` = `lo-blockage` initializes with highest similarity of 0.989 from time 1.0. Thus, although this cycle began with the single model for `drain` = `normal`, it ends by discarding that model and replacing it with two tested fault models: `drain` = `hi-blockage` and `drain` = `lo-blockage`.

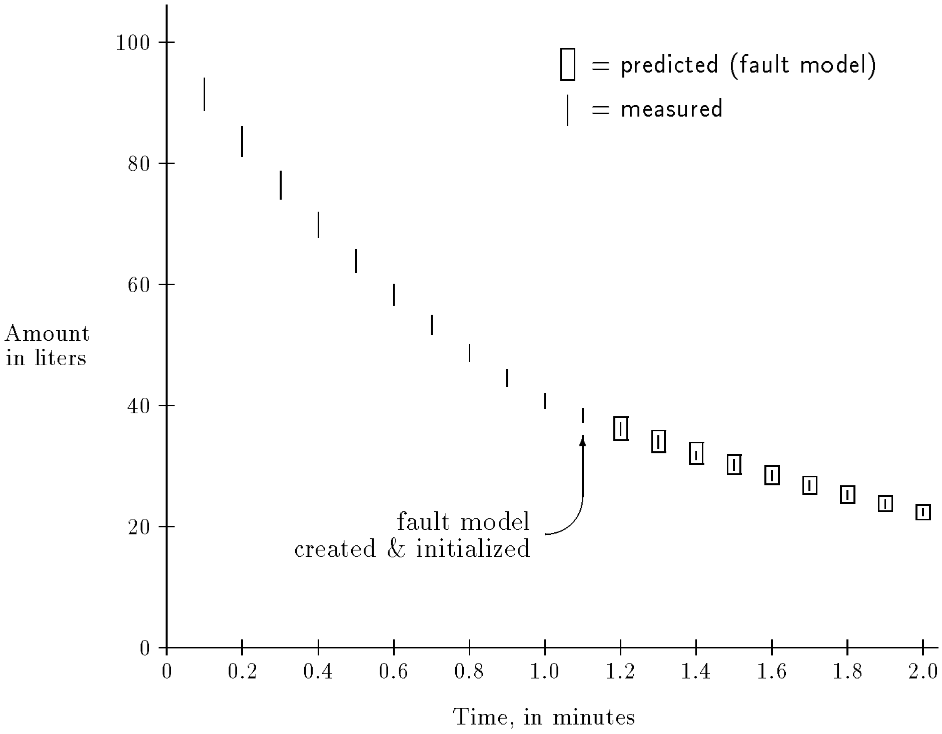
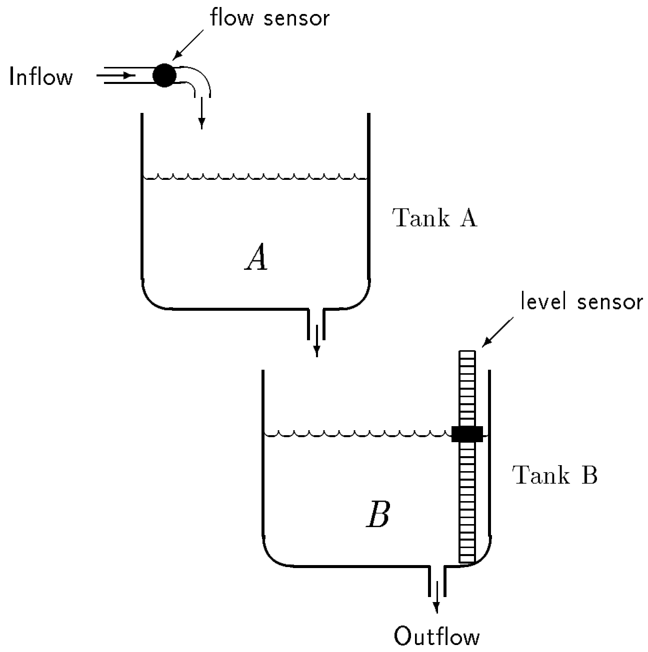


Figure 4.6: Gravity-flow tank: measurements and predictions. After a fault was detected at $t = 1.1$, a fault model was initialized for `Drain` = `lo-blockage`. This graph shows the fault model's agreement with the subsequent measurements.

4.1.3 Cycle 12: t = 1.2

Hypothesis discrimination The measurements that arrive at the beginning of each cycle serve as a test for all models in the tracking set. In this cycle, the predictions from the model `drain` = `lo-blockage` pass all discrepancy tests, but the predictions from the model `drain` = `hi-blockage` fail the trend test. The subsequent effort to resolve this discrepancy fails, so the model is refuted. No attempt is made to generate hypotheses from this discrepancy because there is another model in the tracking set that strongly matches the current measurements (as described earlier in section 3.5.1).

Corroboration After a fault model is created and successfully initialized, it is placed in the tracking set. The model survives as long as its predictions are corroborated by subsequent measurements. As Figure 4.6 shows, the model for `Drain` = `lo-blockage` was created and initialized at $t = 1.1$ and then tested by subsequent measurements. This model emerged as the sole surviving hypothesis.



Differential equations:

$A' = inflow - f(A)$

$B' = f(A) - g(B)$

Figure 4.7: Two-tank cascade. Water flows into Tank-A at a measured rate, which then drains into Tank-B, which drains to the outside. The level of water in Tank-B is measured, but the level in Tank-A is not.

4.1.4 Cycle 22: t = 2.2

Semiquantitative simulation On each monitoring cycle MIMIC uses two different methods of semiquantitative simulation (Q2 and NSIM) in order to obtain the tightest possible bounds on predicted values. In all cycles so far, the tightest predictions have been generated by NSIM. However, the last cycle in the execution history, at $t = 2.2$, shows an exception. NSIM predicted a range of [18.81 20.79] for `Amount` and Q2 predicted a range of [19.00 20.88]. Thus, in this cycle, the greatest lower-bound came from Q2 (19.00) and the least upper-bound came from NSIM (20.79).

4.2 Two-Tank Cascade

The two-tank cascade, as shown in Figure 4.7, represents the next step up in complexity from the gravity-flow tank. Here, there are two state variables, `Amount-A` and `Amount-B`, where `Amount-B` is affected by `Amount-A`. The semiquantitative model, as shown in Figure 4.8, exhibits a more complex dynamic behavior than the gravity-flow tank (see Figure 4.9). Since there is no sensor to measure `Amount-A` in this example, and since an obstructed drain in Tank-A could lead to overflow, the early warning capability of MIMIC becomes important.

```
(define-QDE TWO-TANK-CASCADE
  (quantity-spaces
    (Inflow-A    (0 normal inf)     "flow(out->A)")
    (Amount-A    (0 full)           "amount(A)")
    (Outflow-A   (0 max)            "flow(A->B)")
    (Netflow-A   (minf 0 inf)       "d amount(A)")
    (Amount-B    (0 full)           "amount(B)")
    (Outflow-B   (0 max)            "flow(B->out)")
    (Netflow-B   (minf 0 inf)       "d amount(B)" )
    (Drain-A     (0 vlo lo normal))
    (Drain-B     (0 vlo lo normal)))

  (constraints
    ((MULT Amount-A Drain-A Outflow-A)   (full normal max))
    ((ADD  Outflow-A Netflow-A Inflow-A))
    ((D/DT Amount-A Netflow-A))
    ((MULT Amount-B Drain-B Outflow-B)   (full normal max))
    ((ADD  Outflow-B Netflow-B Outflow-A)) ; Outflow-A = Inflow-B
    ((D/DT Amount-B Netflow-B)))

  (independent   Inflow-A Drain-A Drain-B)
  (history       Amount-A Amount-B)
  (unreachable-values
    (netflow-a minf inf) (netflow-b minf inf) (inflow-a inf))
  (initial-ranges
    ((Inflow-A    normal) (3.01   3.19))        ; +/- 3% of 3.1
    ((Amount-A    full)   (99     101))
    ((Amount-B    full)   (99     101))
    ((Time   T0)          (0      0))
    ((Drain-A normal)     (0.0485 0.0515))      ; +/- 3% of .05
    ((Drain-B normal)     (0.0485 0.0515))      ; +/- 3% of .05
    ((Drain-A lo)         (0.020  0.0485))
    ((Drain-B lo)         (0.020  0.0485))
    ((Drain-A vlo)        (0      0.020))
    ((Drain-B vlo)        (0      0.020))))
```

Figure 4.8: Semiquantitative model of the two-tank cascade. This model is the next step up in complexity from the gravity-flow tank because it contains two state variables, `Amount-A` and `Amount-B`, where `Amount-B` is affected by `Amount-A`, and `Amount-A` is not observable.

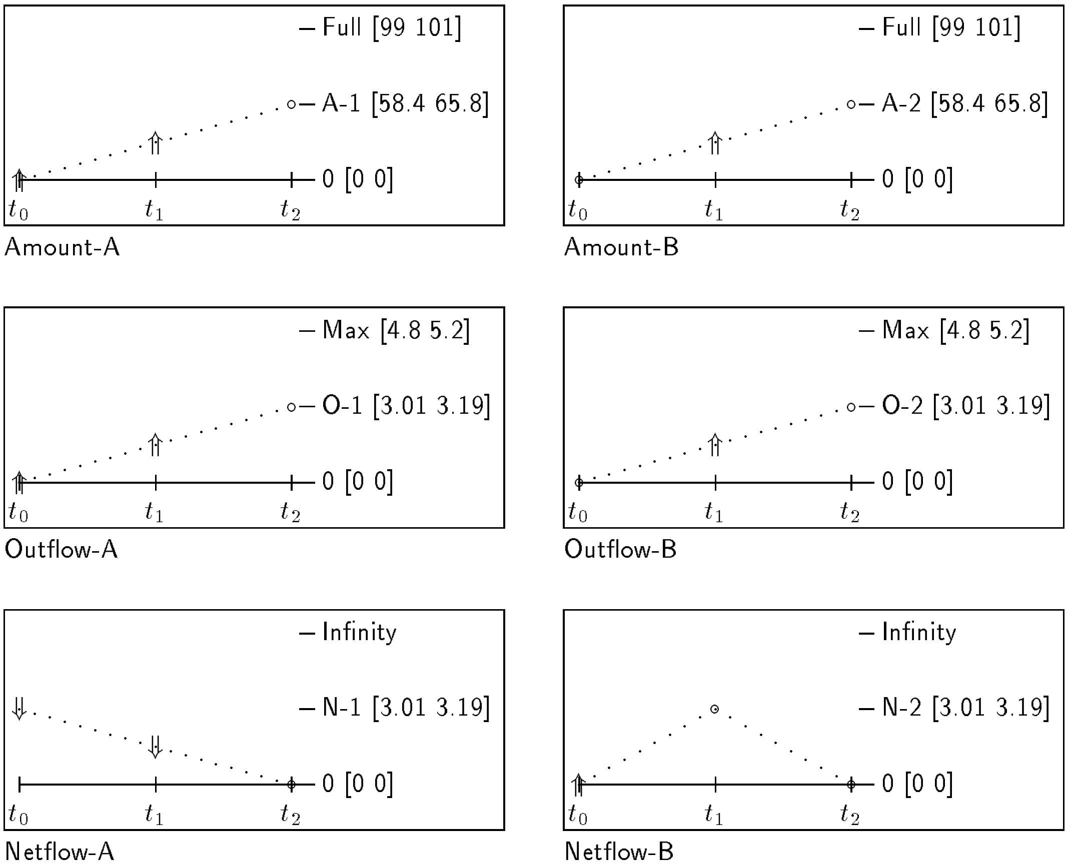


Figure 4.9: Semi-quantitative behavior of a two-tank cascade filling from empty, with no faults. In contrast with the previous example, inflow is greater than zero. Time point t_0 represents the starting time and t_2 represents the time when equilibrium is reached. Time t_1 identifies the time at which Netflow-B changes from increasing to decreasing. The ranges associated with each landmark value, such as A-1 [58.4 65.8], are guaranteed to bound the correct value.

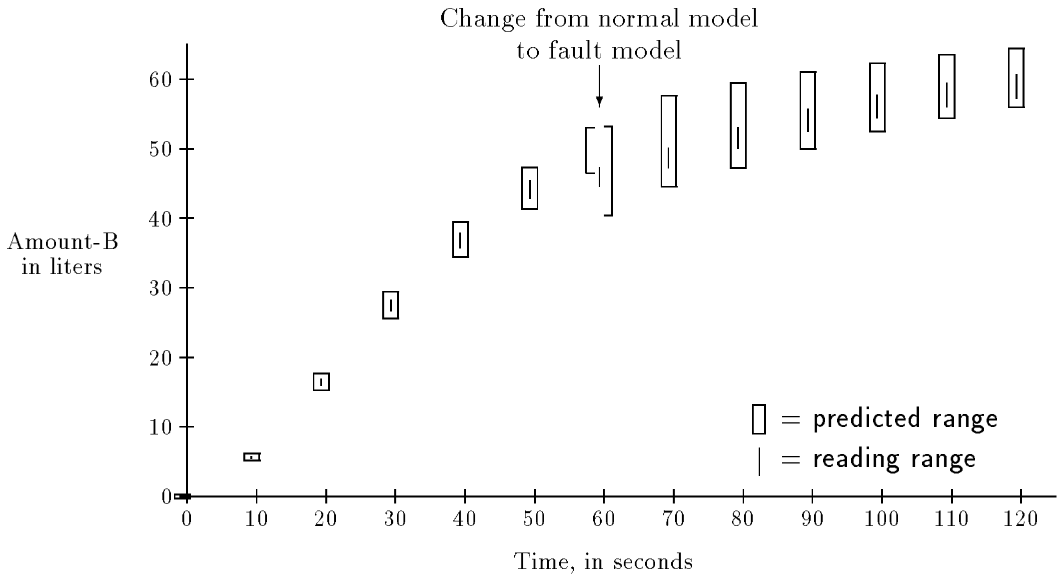


Figure 4.10: Sensor readings and predictions for the two-tank cascade, with a fault at t = 50.01. The normal model is tracked up to t = 60, when an analytical discrepancy is detected. From that discrepancy the hypothesis `Drain-A = lo` is proposed and the corresponding fault model is initialized. Subsequent readings show agreement with the fault model's predictions. The reason why the predicted ranges are larger with the fault model than with the normal model is because the range associated with `Drain-A = lo` [.02 .0485] is larger than the range associated with `Drain-A = normal` [.0485 .0515].

The monitoring scenario for the two-tank cascade begins with both tanks empty when filling begins with a constant inflow. At t = 50.01, just after the measurement at t = 50, the drain for Tank-A becomes partially clogged. The highlights from MIMIC's execution history are summarized below; the history is not included in the appendix.

1. Before the failure at t = 50.01 occurs, Mimic first has to track past an inflection in the qdir of `Netflow-B`. The readings for t = 20, 30, and 40 show that the second derivative of `Amount-B` decreases from about 0.050 to 0.002 to -0.017. A discrepancy is detected at t = 40 since a second derivative of -0.017 is incompatible with `Netflow-B`'s predicted qdir of `inc`. MIMIC resolves this discrepancy by finding the successor state to be compatible.

2. At t = 50 the single state being tracked from the normal model has a very strong similarity to the readings (0.977). The partial clog in the upper drain then occurs at t = 50.01.

3. At t = 60 the single state remains compatible with readings, but just barely; the observed value of `Amount-B` barely overlaps the predicted range (see Figure 4.10).

Similarity has dropped to 0.181, but in updating the state of the model from the readings, an analytical discrepancy is detected by Q2. Thus, although the readings are individually compatible with predictions, propagating these readings through the model's equations reveals that the readings, parameters, and equations are mutually incompatible.

4. The discrepancy with `Amount-B` at $t = 60$ prompts dependency tracing which results in two hypotheses: `Drain-A` = `lo` and `Drain-B` = `lo`. The new-hypothesis state for `Drain-A` = `lo` results in 3 completions differing only in the qdir of `Netflow-B`. After resimulation, the estimated time-of-fault for each of these 3 states is $t = 50$, which is "correct" since the failure occurred closer to $t = 50$ than to $t = 60$.

   The new-hypothesis state for `Drain-B` = `lo` results in 9 completions differing not only in the qdir of `Netflow-B` but also in the qmag. However, none of these states survives resimulation because, in all cases, the upper bound of the reading for `Amount-B` at $t = 60$ is *less than* the lower bound predicted by Q2/NSIM. Thus, the only surviving candidate at $t = 60$ is `Drain-A` = `lo`, with 3 states.

5. At $t = 70$ two of the three states for candidate `Drain-A` = `lo` have a discrepancy with the qdir of `Netflow-B`. These two states are discarded without any attempt to form new hypotheses because the candidate still has a valid state.

6. At $t = 90$, MIMIC has to get past the exact same inflection in `Netflow-B`'s qdir that it encountered at $t = 40$. The reason that this has occurred again is because the failure at $t = 50.01$ effectively set us back to an earlier position in the qualitative behavior of the two-tank cascade. The three new-hypothesis states that were created at $t = 60$ for this candidate actually represented all three possibilities with respect to the inflection point (before, at, and after), but only the "before inflection" state survived to this point.

   There are seven successors to the state discarded at $t = 90$, but only two of them are found to be compatible with the readings. Thus, the cycle at $t = 90$ ends with one model having two behaviors.

7. In all remaining readings, the two behaviors of candidate `Drain-A = lo` survive. The two behaviors differ from each other in the value for `Inflow-A` because when they were created at $t = 90$, the lower-bound of `Inflow-A` (an independent variable) was tightened from its initial value of 3.01 to 3.06 due to Q2 propagation of readings at $t = 90$.

4.3 Open-Ended U-Tube

The open-ended U-tube, as shown in Figure 4.11, is more complex than the two-tank cascade because the two tanks affect each other, whereas in the two-tank cascade, the

Differential equations:

$$A' = inflow - f(A, B)$$
$$B' = f(A, B) - g(B)$$

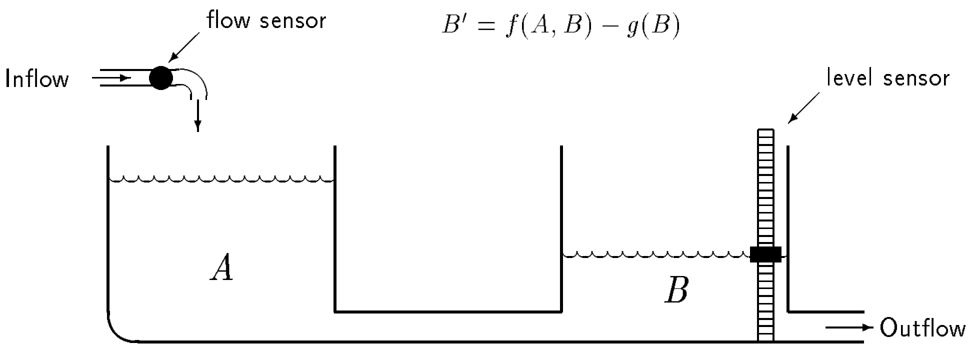


Figure 4.11: Open-ended U-tube. Water flows into Tank-A at a measured rate, which then flows into Tank-B through the connecting pipe. The water in Tank-B drains to the outside. The level of water in Tank-B is measured, but the level in Tank-A is not.

downstream tank does not affect the upstream tank. The semiquantitative model is shown in Figure 4.12. Experiments with this example motivated the development of the acceleration test for discrepancy-detection because a particular [subtle] fault was undetectable with the limit and trend tests.

The monitoring scenario for the open-ended U-tube begins with both tanks empty when a constant inflow is applied to Tank-A. At $t = 50.01$, just after the measurement at $t = 50$, a partial clog develops in the pipe connecting the two tanks. The effect of this fault would be most visible in measuring the amount in Tank-A, but only Tank-B has an amount sensor in this example. The highlights from MIMIC's execution history are summarized below; the history is not included in the appendix.

1. At $t = 10$ the first sensor readings appear and are tracked.

2. At $t = 20$ MIMIC tracks past an inflection in the qdir of `Netflow-b`.

3. At $t = 50.01$ a partial clog develops in the pipe, but no discrepancy is detected with the following reading at $t = 60$, although the similarity drops markedly.

4. At $t = 70$ an acceleration discrepancy is detected. Why wasn't this discrepancy detected at $t = 60$? If the deviation had been larger, this would have been possible, but the deviation was small and the readings at $t = 70$ were the first chance to compute a second derivative based on two consecutive first derivatives following the failure.

   Two hypotheses are formed at this time: `Conductance` = `lo`, with 5 completions, and `Drain-B` = `lo`, with 15 completions. The high number of completions is a result of

```
(define-QDE OPEN-U-TUBE
  (quantity-spaces
    (Inflow-A    (0 normal inf))
    (Amount-A    (0 full))
    (Pressure-A  (0 max))
    (Pressure-B  (0 max))
    (P-diff      (minf 0 inf))
    (Flow-AB     (minf 0 inf))
    (Netflow-A   (minf 0 inf))
    (Amount-B    (0 full))
    (Outflow-B   (0 max))
    (Netflow-B   (minf 0 inf))
    (Drain-B     (0 vlo lo normal))
    (K-A         (0 KA*))
    (K-B         (0 KB*))
    (Conductance (0 vlo lo normal)))

  (constraints
    ((MULT  Amount-A  K-A  Pressure-A)       (full KA* max))
    ((MULT  Amount-B  K-B  Pressure-B)       (full KB* max))
    ((ADD   P-diff  Pressure-B  Pressure-A))
    ((MULT  P-diff  Conductance  Flow-AB))
    ((ADD   Netflow-A  Flow-AB  Inflow-A))     ; Netflow = Inflow - Outflow
    ((D/DT  Amount-A Netflow-A))
    ((MULT  Amount-B  Drain-B  Outflow-B)      (full normal max))
    ((ADD   Netflow-B  Outflow-B  Flow-AB))    ; Netflow = Inflow - Outflow
    ((D/DT  Amount-B Netflow-B)))

  (independent Inflow-A K-A K-B Conductance Drain-B)
  (history     Amount-A Amount-B)
  (unreachable-values (netflow-a minf inf) (netflow-b minf inf)
      (inflow-a inf) (P-diff minf inf) (flow-AB minf inf))

  (initial-ranges ((Drain-B normal) (0.0485 0.0515)) ; +/- 3% of .05
  ((Drain-B lo)     (0.020 0.0485))
  ((Drain-B vlo)    (0     0.020))
  ((Inflow-A    normal) (3.01  3.19))
  ((Amount-A    full)   (99    101))
  ((Amount-B    full)   (99    101))
  ((K-A         KA*)    (0.97  1.03))
  ((K-B         KB*)    (0.97  1.03))
  ((Conductance normal) (.2425 .2575))
  ((Conductance lo)     (.125  .2425))
  ((Conductance vlo)    (0     .125))
  ((Time   T0)          (0     0))))
```

Figure 4.12: Semiquantitative model of the open-ended U-tube.

two things: the lack of any readings for `Amount-A` and the [necessarily] conservative approach that MIMIC takes in determining initial values for a new hypothesis. zzzz

5. At $t = 80$, 2 of the 20 states being tracked exhibit an acceleration discrepancy, and in each case tracking finds two compatible successor states. From $t = 90$ until the end, a total of 22 states are being tracked.

6. At the end, at $t = 110$, the two candidates still remain, with 22 states total. The correct candidate (`Conductance = lo`) has a significantly higher similarity than the other candidate (0.753 vs. 0.343). The similarity of the incorrect candidate (`Drain-B = lo`) has steadily decreased from its initial similarity of 0.531 to 0.343, but it has remained compatible with observations because: (a) not enough time has elapsed for it to be detected by the limit test or analytical test, and (b) the near-equilibrium state of the U-tube has not provided the kinds of dynamic changes in behavior that the trend and acceleration tests depend on for discrepancy detection.

   The multiple states of a given candidate are due to differences between `Netflow-A` and `Netflow-B`, usually due to qdir differences. Since there are no readings for `Amount-A` or `Netflow-A`, MIMIC cannot detect any discrepancies with `Amount-A` or its derivatives, so it must carry along all these states. Recent work by Fouché [Fou91] on aggregating such "irrelevant distinctions" apply to this problem and may help substantially. Importantly, the MIMIC framework allows this to be factored out as a distinct problem, so independent advances on this problem will benefit MIMIC.

4.4 Vacuum Chamber

The most detailed practical example of fault detection using MIMIC is given in Kay's thesis on the dynamic envelope method [Kay91], as described earlier in section 3.3.4 and implemented in NSIM. Kay illustrates the method through monitoring and diagnosis of a vacuum chamber using an early version of MIMIC. The vacuum chamber presents a compelling application for monitoring, as Kay explains:

> The production of high vacuum is of great importance to semiconductor fabrication as many of the steps (such as sputtering and molecular beam epitaxy) cannot be performed if there are foreign particles in the process chamber. Unfortunately, creating such ultra-high vacua can be expensive and time-consuming. To reach ultimate pressures of 10^{-9} torr[1] can take several hours and something as innocuous as a fingerprint left on the chamber during servicing can cause a huge performance loss. Because of this risk, it is important to service vacuum

[1] Torr (from Torricelli) is a unit of pressure equal to 1333.22 microbars, the pressure needed to support a column of mercury one millimeter high under standard conditions.

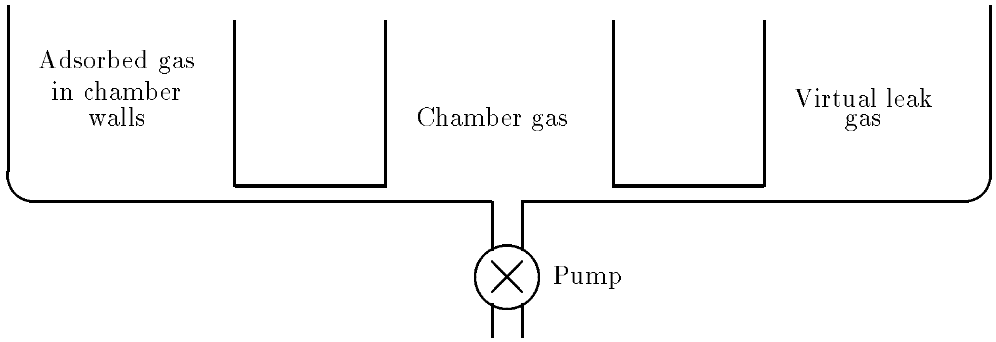


Figure 4.13: Vacuum chamber and vacuum pump. Gas is removed from the main chamber by pumping action; the adsorbed gas on the chamber walls "outgasses" as the pressure drops and is then pumped out. This 3-tank model allows for a "virtual leak", which is present in some fault models.

> equipment only when there is a problem. This suggests a need for a monitoring system that can detect when the system goes out of tolerance. Of particular importance is the time during which the chamber is being pumped down from atmospheric pressure. If failures during this fairly short period (around 15 to 30 minutes) can be detected, much time and expense can be avoided.

The vacuum chamber is a particularly appropriate application for semiquantitative modeling and simulation because there is no practical theory for the sorption of gases. Specifically, the adsorption and desorption of gas from chamber walls is not understood precisely, so the model must account for this incomplete state of knowledge.

Kay modeled the vacuum chamber as a U-tube in which one compartment contains the relatively large amount of chamber gas and the other compartment contains the relatively small amount of gas adsorbed on chamber walls (see Figure 4.13). The chamber gas obeys the ideal gas law ($PV = nRT$), but the adsorbed gas — or more specifically, its rate of adsorption and desorption — can only be approximated. Likewise, pump performance is approximated as a function of chamber pressure. Suffice it to say that there are several uncertain values and relations in the model, so the ability to express these uncertainties directly in the simulation model is valuable.

The earliest version of MIMIC, which Kay used in his experiments, used only Q2 for semiquantitative prediction and did not yet incorporate the technique of state-insertion for measurements or other discrepancy-detection tests besides the limit test. Thus, Kay's tests primarily illustrate the power of dynamic envelopes (NSIM) to detect abnormal behavior over long time intervals. In one test, MIMIC monitored measurements from a simulated vacuum chamber with a gasket leak of about 1.6 ml/sec. During the approximately 30-minute pump-down phase, MIMIC *without* dynamic envelopes detected the fault after 9 minutes, but *with* dynamic envelopes it was able to detect the fault in only 4 minutes. Thus, Kay's experiment demonstrates the power of dynamic envelopes in semiquantitative simulation *and* it also demonstrates a practical application of MIMIC in a significant industrial process.

4.5 The Dynamics Debate

Critics may point out, correctly, that the four models presented in this chapter do not exhibit "interesting dynamics". The four systems are all first-order and mostly linear (the vacuum chamber contains some non-linear functions), and therefore do not represent difficult problems for scientists and engineers. They argue that useful research addresses only open problems, i.e., problems that scientists don't know how to do or find so difficult that it really isn't done.

An alternate view, best expressed by Brian Falkenhainer and Johan de Kleer in a recent network discussion, is that useful research in Qualitative Physics encompasses a wider range of issues. Scientists and engineers form a relatively small class of people; there is a far larger class of people who have to operate, troubleshoot, and repair our technology. And there is probably more payoff to society in making these people more productive. One goal of Qualitative Physics is to build the technology needed for design and diagnosis tools to enable the large class of technicians to do their jobs better.

Towards this goal, the research issues aren't so much in the dynamics of the system but in the surrounding machinery. What are the sound inferences that can be drawn from partial information? What kind of abstractions should be used in modeling the system? Which faulty component may be causing the observed abnormality? Does the model over-estimate or under-estimate? There are many "simple" systems out there which we don't know how to diagnose efficiently. This dissertation speaks to the surrounding machinery—the machinery that enables monitoring and diagnosis based on sound inference from partial knowledge of a mechanism.

Chapter 5

Discussion and Conclusions

As a one-time engineer, Mott loved that word elegant, *for it implied an entire scale of values: an elegant solution had to be simpler than its adversaries, it had to be easily assembled, it had to be cost-efficient, and it had to be instinctively satisfying to the engineering mind.*

Space, by James A. Michener [Mic82]

This dissertation has presented a novel design for monitoring and diagnosis of continuous-variable dynamic systems. This final chapter examines the design to identify its main principles and the benefits that ensue. The importance of the two technological foundations — semiquantitative simulation and model-based diagnosis — become apparent in the description of the design principles and strengths. The chapter ends with directions for future research.

5.1 Design Principles

The design of MIMIC has evolved over time, driven by a growing understanding of the problem of monitoring & diagnosis and by an increasing recognition of the capabilities of semiquantitative simulation and model-based diagnosis. The resulting design embodies a number of principles, some that guided the design from the beginning and some that were recognized in retrospect. The list of principles described below should be combined with the following section on strengths to gain a complete picture of the design's rationale and benefits.

1. *Shape inference in terms of model construction.*
   Clancey's observation at the beginning of chapter 3 is telling — a diagnosis should describe what is happening in the world, causally relating states and processes. His suggestion to consider inference in terms of model construction requires a model of the world — a model that can be modified to reflect alternate hypotheses about the world. The purpose of diagnosis then becomes one of constructing the right model and verifying its agreement with the world. In MIMIC this is seen in the paradigms of "monitoring as model corroboration" and "diagnosis as model modification".

2. *Integrate monitoring and diagnosis.*
   The tasks of monitoring and diagnosis are integrated in MIMIC in a natural way because they work on *exactly* the same information: the models and behaviors in the tracking set. The same method for predicting normal behavior limits during monitoring is used again to predict fault behavior during diagnosis. The same discrepancy-detection methods used to detect the initial problem during monitoring are used again to discriminate among the resulting hypotheses during diagnosis. The same similarity metric used to rank hypotheses during monitoring is used again to estimate the time-of-fault during the resimulation phase of diagnosis. The same warning predicates that are used during monitoring to detect undesirable behavior are used again to forewarn of risk during diagnosis. In short, MIMIC's design views monitoring and diagnosis as two sides of the same coin, and this view enables an elegant sharing of techniques.

3. *Exploit dynamic behavior for clues.*
   Observations of dynamic behavior over time provide stronger clues about faults than can be obtained from a single snapshot of system output. This may seem obvious, but few fault detection schemes actually base their conclusions on a *sequence* of observations. Most approaches to automated diagnosis have focused on maintenance diagnosis, where all the effects of a fault have propagated and the system is off-line. In operative diagnosis, however, the effects are still propagating and the system remains in operation. The primary source of information in operative diagnosis is a stream of sensor readings, and MIMIC uses this information to the fullest. This means comparing the time-varying observations to time-varying predictions, seeking to corroborate or refute behaviors that follow from the fault hypotheses, and updating the state of a model with new measurements.

4. *Use semiquantitative simulation, not heuristics.*
   Years of experience with first-generation expert systems have revealed serious weaknesses in their ability to reliably detect and diagnose faults. The ways in which a single fault can manifest are numerous, depending on the severity of the fault and the state of the system. A fault does not necessarily trip the same alarms each time, nor does it always trip them in the same sequence. Add to this the possibility that multiple faults may interact and the fact that many large systems customarily operate with at least one fault, and the limitations of the heuristic approach become clear.

   Given a system with complex interactions and tight coupling, the only practical way to predict the possible manifestations of a fault is to simulate the fault model in a way that reveals all behaviors consistent with incomplete knowledge of the mechanism and its observed state. Semiquantitative simulation is the enabling technology that makes this possible. Then, through model-based diagnosis we can infer the possible faults — all without resort to experience-based heuristics.

5. *Discrepancy-detection is key.*
The ability to detect discrepancies between measurements and predictions is the single most important factor in controlling the complexity of monitoring and diagnosis. This ability depends on well-placed sensors, adequate model precision, and sensitive discrepancy-detection methods. The need for precision motivated the inclusion of the NSIM dynamic envelope method and the Q3 technique of state-insertion for measurements. Similarly, the need for sensitive discrepancy-detection methods drove the development of the trend, acceleration, and analytical tests — all aimed at extracting the most possible information from model and measurements. The continuing challenge is to discover additional methods in this open-ended category that are conservative (no false positives) yet sensitive.

5.2 Strengths

1. *Expressive Power*
Semiquantitative models provide greater expressive power for states of incomplete knowledge than differential equations, and thus make it possible to build models without incorporating assumptions of linearity or specific values for incompletely known constants. The modeler can express incomplete knowledge of parameter values and monotonic functional relationships (both linear and non-linear). By specifying conservative ranges for landmark values and conservative envelope functions for monotonic relationships, semiquantitative simulation generates guaranteed bounds for all state variables. This eliminates modeling approximations and compromises as a source of false positives during diagnosis.

2. *Soundness*
Qualitative simulation generates all possible behaviors of the mechanism that are consistent with the incomplete/imprecise knowledge. This is essential for distinguishing *misbehavior* (which is due to a fault, and thus requires diagnosis) from normal behavior, especially when there is more than one possible normal behavior. This eliminates the "missing prediction error" as a source of false positives during diagnosis.

3. *Early Warning*
Given the set of hypothesized models in the tracking set, MIMIC can simulate these models ahead in *qualitative* time to efficiently predict the possible futures and forewarn of imminent harm (risk). Similarly, the effects of proposed control actions can be determined by simulating from the current state — a valuable capability in complex systems. Both features take advantage of semiquantitative simulation's ability to predict all possible behaviors consistent with an incomplete state of knowledge.

4. *Dynamic Alarm Thresholds*
Incremental simulation of the semiquantitative model in synchrony with incoming sensor readings generates, in effect, dynamically changing alarm thresholds. Comparison

of observations to model predictions permits earlier fault detection than with fixed-threshold alarms and eliminates false alarms during periods of significant dynamic change, such as startup and shutdown.

5. *Temporal Uncertainty*
   Because a given fault may manifest in different ways under different circumstances, methods that identify faults based on specific subsets of alarms or specifically-ordered sequences of alarms are insufficient. Since MIMIC matches observations against a branching-time description of predicted behavior (a description that includes all valid orderings of events), and since it tests for overlap of uncertain value ranges rather than whether or not an alarm is active, it can detect all of the valid ways in which a fault manifests.

6. *Graceful Degradation*
   Large mechanisms incorporating fault-tolerant design tend to degrade gracefully as unrepaired faults accumulate. Similarly, MIMIC's predictions of faulty behavior tend to degrade (i.e., to be less precise, but still sound) as more faults are incorporated into a hypothesis. Such graceful degradation is preferable to the problem of "falling off the knowledge cliff" — the problem that occurs with heuristic-based systems when the combination of faults surpasses the system's ability to diagnose.

7. *Cognitive Support*

> Design engineers and, to a lesser extent, plant operators, can reasonably well anticipate the pattern of alarms that will be triggered by a known malfunction. To a lesser degree, this is the case even for multiple, simultaneous malfunctions. On the other hand, they have great difficulty reasoning in the reverse direction: that is, mapping from a complicated pattern of sensor alarms back to causative faults. [Mal87]

Like any other operator advisory system, MIMIC was designed from the outset to assist the operator in the difficult task of diagnosis. However, unlike the methods in current industrial practice, MIMIC forms a fault-model that accounts for the observations over time. As a result, MIMIC can provide a more satisfying explanation of its hypothesis than a symptom-fault association based on experience; MIMIC can show that its hypothesis has accurately predicted the observed behavior over the last n measurements. MIMIC's model is also more *informative* because it shows the values of unseen variables and predicts future consequences. Thus, MIMIC not only relieves some of the cognitive load but also provides results in a form that makes justifiable sense to the operator.

5.3 Limitations

This section on limitations and the following section on future work should be considered as parts of a larger whole. To a great extent, the limitations listed here are areas for future work, and the areas for future work reveal limitations of the current design.

5.3.1 Temporal Abstraction

In the current work, MIMIC uses a behavioral model based on qualitative differential equations, in the style of QSIM. Such a model gives predictions for each variable of magnitude and direction-of-change for time points and time intervals. This kind of description of dynamic behavior is very useful in detecting and diagnosing faults, as shown in Chapter 4, but is undesirably detailed for some troubleshooting scenarios. Hamscher has emphasized this point, showing the value of using a representation that makes explicit the behavior at *a high level of temporal abstraction* [Ham91]. For example, although an instruction-level simulation of a microprocessor explicitly represents the logic levels present on the external bus at every clock edge, it does not reveal the simple fact that during normal operation those bus signals should be very active.

One form of temporal abstraction is achieved by shifting from the time domain to the frequency domain, but Hamscher's example uses an even more abstract representation: a signal is either changing or constant. This aspect of a signal is easy to observe and still supports effective troubleshooting. Similarly, as we saw in Chapter 2, KARDIO uses a temporal abstraction that describes a signal as having a regular or irregular rhythm, and a model that determines that the sum of two signals — one with regular and one with irregular rhythm — is a signal with irregular rhythm.

The use of temporal abstraction still fits within the hypothesize-build-simulate-match architecture of MIMIC; it just requires an appropriate model. However, in any modeling language that might be created to express temporal abstractions, we want to preserve two key properties that QSIM provides: (1) the ability to predict all behaviors that are consistent with the incomplete knowledge, and (2) the ability to bound the predictions based on the incomplete quantitative knowledge. This task — that of creating a modeling formalism for various types of temporal abstraction — is an important area for future work.

5.3.2 Dependency Tracing

MIMIC currently uses a graph traversal procedure to trace upstream from discrepancies to identify suspects in the structural model of the mechanism. While this method is intuitively understandable and identifies all appropriate suspects, it can also identify some unnecessary suspects. For example, given a multiplier that computes $x \times y = z$ and the discrepancy $z = 0$ and the observation $x \neq 0$, the fault must either be in the multiplier or upstream of y; it cannot be upstream of x because no non-zero value of x can cause $z = 0$

(assuming a functioning multiplier). Dependency tracing, however, will trace upstream of x because it treats the multiplier as a black box.

A better technique for identifying suspects is to have the simulator record dependencies for each predicted value (SOPHIE [BBd82] provided one of the earliest examples of this approach). Thus, when a discrepancy is detected for variable z, the set of components and parameters that participated in z's predicted value are available in the dependency trail. This basic idea is used in assumption-based truth maintenance [dKW87].

5.3.3 Spurious Behaviors

Qualitative simulation can predict behaviors that do not correspond to the solution of any ODE covered by the QDE. These are called *spurious behaviors* because they do not appear in any real mechanism that the QDE represents. Some spurious behaviors are eliminated by global filters that check specific mathematical or physical properties of the behavior, such as the analytic functions filter, the curvature-at-steady filter, and the energy filter. However, some spurious behaviors may still survive.

Spurious behaviors can be a source of false negatives during fault detection because a faulty behavior observed in the real mechanism might coincidentally match a spurious behavior. Thus, misbehavior could go undetected. The solution, of course, is to eliminate spurious behavior predictions through the discovery of additional global filters which are both mathematically and physically valid. This requires more research into the sources of incompleteness that leads to spurious behaviors.

5.3.4 Cascading Faults

The hypothesis-generation algorithm of MIMIC makes the simplifying assumption that new symptoms are due to a *single* new fault or repair. For mechanisms that are monitored with frequent periodic sensor readings, this is usually a correct assumption. However, there is a class of faults known as *cascading faults* or *consequential faults* in which one fault may cause another fault almost immediately, and another, and another. If the symptoms of two faults (A and B) initially appear at time t, MIMIC will generate single-change hypotheses which include $\{A\}$ and $\{B\}$ individually, but not $\{A, B\}$ together. Unless symptoms of A or B persist until time $t+1$ or reappear at some future time, MIMIC will fail to diagnose the combination $\{A, B\}$.

There are two approaches to the multiple-simultaneous-fault problem. The simplest is to permit generation of multiple-change hypotheses up to some value N, where N is the maximum number of multiple changes. This approach is likely to ensure that the correct hypothesis is generated, but it substantially increases the size of the hypothesis space every time hypotheses are generated. An alternate approach is to exploit domain-specific knowledge of cascading faults. For example, if it is known that fault A usually causes an immediate fault B, then it is straightforward to hypothesize $\{A\}$ and $\{A, B\}$. This approach is much more computationally tractable, but is also vulnerable to gaps in the domain-specific knowledge.

5.3.5 Complexity and Real-Time Performance

The MIMIC design makes no guarantees about real-time performance because the amount of work to be done on each cycle of the main loop is not predictable. As we noted in Chapter 3, there is exponential complexity in behavior generation and hypothesis generation. Although this complexity is largely controlled through practical methods and simplifying assumptions, the MIMIC algorithm does not impose any upper limits on the amount of computation that may be consumed during behavior generation or hypothesis generation.

5.4 Appropriate Domains

For what types of applications is the MIMIC design appropriate? As noted at the beginning of this dissertation, MIMIC is intended for deterministic continuous-time dynamic systems that must be continually monitored from sensor readings and diagnosed during system operation. This description applies to much of the process industries (chemical processing, power generation, food processing, etc.) as well as medical intensive care. Of course, there are methods already in place in these settings, and some of the settings impose demands that MIMIC does not meet. In particular, MIMIC is:

- not appropriate for hard real-time applications where reaching a diagnosis after a deadline is as bad as an incorrect diagnosis;
- not appropriate in applications where multiple simultaneous faults or rapidly cascading faults are common; and
- not appropriate in applications where the most important clues are sensed only by humans, such as through sight, sound, and smell.

Outside of these inappropriate settings, MIMIC is most appropriate in settings where existing methods are inadequate in an area where MIMIC provides an advantage. In particular, MIMIC is:

- appropriate where the mechanism is highly dynamic and therefore difficult to diagnose with existing methods;
- appropriate where false positives are a problem during fault detection;
- appropriate where early warning of the possible consequences of a fault are important;

- appropriate where the failure to consider some hypothesis could be disastrous; and
- appropriate where a precise theory is not available to predict behavior.

5.5 Future Work

5.5.1 Discrepancy Detection

Discrepancy-detection is crucially important in MIMIC because it is used not only to detect anomalies and thus trigger hypothesis generation, but also to refute incorrect candidates during tracking. One way to improve discrepancy-detection is to improve the precision of the model's predictions, and this is the goal of two active lines of research. The first is the elimination of spurious behavior predictions in qualitative simulation, thus eliminating a source of false negatives during fault detection. The second line of research is the discovery of increasingly stronger methods of semiquantitative reasoning. By generating tighter bounds on variables and their derivatives, especially in the presence of noise, some misbehaviors will either be detectable sooner or become detectable for the first time.

5.5.2 Perturbation Analysis

MIMIC does not currently exploit some information that is in the model; this is best described through example. Consider a simple one-tank system where the tank is draining toward empty with a partially clogged drain, which MIMIC has already properly diagnosed. Specifically, the model has three possible values for the drain-rate (`normal`, `low`, and `very-low`), and has correctly diagnosed the value as `low`. Suddenly, the drain is cleared and the outflow rises. When MIMIC detects the discrepancy and identifies the drain as a suspect, it will generate hypotheses not only for `normal` but also for `very-low`. Of course, common sense tells us that, given the discrepancy of increased outflow, the drain-rate cannot be less than what it was, i.e., it cannot be `very-low`.

The problem is that MIMIC treats the drain (and every other suspect) as a black box whose fault modes must be hypothesized. If MIMIC knew the sign of the partial derivative of outflow with respect to drain-rate (with everything else remaining the same), then it could eliminate such nonsensical hypotheses. Weld's work on comparative analysis [Wel88a] and exaggeration [Wel88b] provides exactly what is needed — it determines how a mechanism will react to perturbations in a parameter. By adding such perturbation analysis at hypothesis-generation time, MIMIC can reduce the number of hypotheses to be modeled, tracked, and refuted. Further, the same information can be used to determine the initial qdirs of some variables when a new fault model is initialized, thereby reducing the number of initial states. Both capabilities will help control complexity in MIMIC.

5.5.3 Component-Connection Models

MIMIC requires two models of the mechanism being monitored: a behavioral model expressed in qualitative differential equations and a structural model expressed as a network of components and connections. The two models are built separately and care must be taken to ensure that they are consistent with each other. Further, creation of the behavioral model is tedious because it is assembled as a set of equations — a level of description that is unlike the mechanism itself, which is assembled from a set of standard components.

Ideally, a model builder could create a model by assembling it from a library of standard components, in much the same way that the physical mechanism is assembled. In this vein, Franke has developed CC, a model-building program that accepts a component-connection description of a physical system and translates it into the qualitative differential equations of QSIM [FD90]. This permits a domain expert to specify the model in terms that are more natural to the domain (components and connections) and ensures that the behavioral model is always consistent with the structural model. CC provides facilities for component abstraction and hierarchical component definition, raising the level of abstraction for modeling via QSIM. Since a CC model is organized as a set of explicit components and connections, it contains the structural information needed by MIMIC for hypothesis generation. A natural step then, in future work, is to modify the hypothesis-generation code of MIMIC to work directly with a CC model.

5.5.4 Hierarchical Representation and Diagnosis

To handle larger, more complex systems, MIMIC will need to add hierarchical diagnosis. Of course, this requires a hierarchical representation of the mechanism, and such a capability is now available in the CC component-connection language. Hierarchical modeling is independent of fault modeling, of course; fault models may appear at any level of a hierarchy. A diagnostic system that has both has two separate methods for making a diagnosis more specific: *refinement* and *decomposition*. Given a suspect that has passed the test of constraint suspension, refinement is the process of instantiating and testing its fault models. Similarly, given a confirmed suspect, decomposition is the process of moving to a more detailed description of the suspect's individual components in order to achieve a more localized identification of the faulty unit. With both methods, the question arises as to the proper control structure for choosing between refinement and decomposition. Hamscher's diagnostic engine, XDE, always prefers refinement over decomposition, as shown in Figure 5.1. However, he states that this is not ideal:

> Refinement has priority over decomposition because, again heuristically, refinement is often able to rule out alternative diagnoses while decomposition often increases the number of alternatives. Experience with XDE suggests the need for research into a more flexible control structure. The program should not always

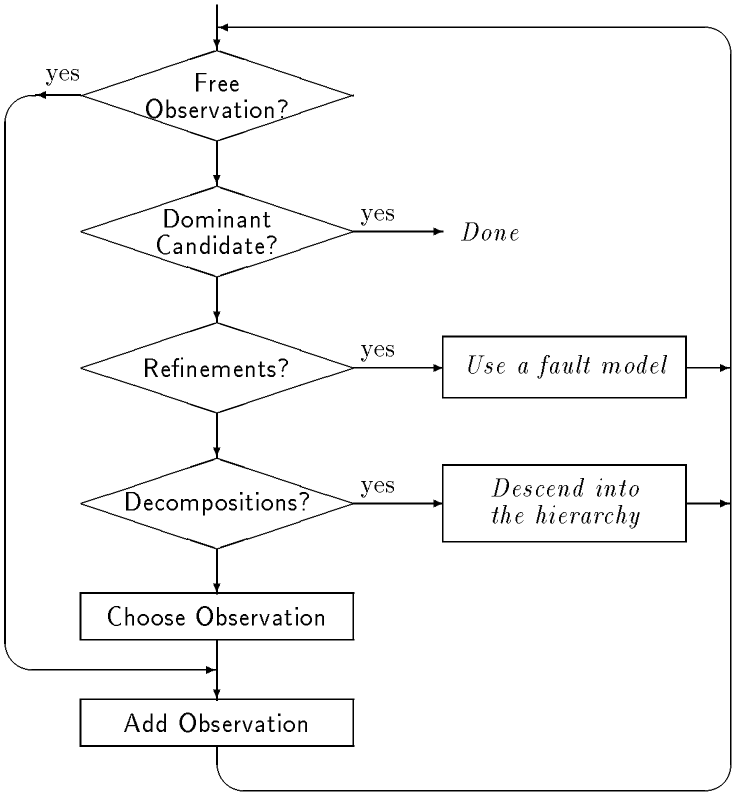


Figure 5.1: Control flow in XDE currently prefers refinement over decomposition. Hamscher believes that the control can be improved by basing the decision of refinement vs. decomposition vs. probes on the current state of the diagnosis.

> try refinements before decompositions, nor decompositions before probes; people clearly make use of the current state of the diagnosis to make this decision, and so should the program. [Ham91]

Future work in this area should be guided by Hamscher's insight.

5.5.5 Scale-Space Filtering

Noisy signals are a fact of life in process monitoring, but effective monitoring and diagnosis depend on either filtering out the noise or somehow accounting for noise when reasoning with noisy sensor readings. One common method treats noise as a high-frequency component on a low-frequency signal, and therefore removes noise using a low-pass filter. Unfortunately, this method also removes transients indicative of certain faults. Another common method models noise as a zero-mean gaussian function of known variance, as in the Kalman filter; the effect of noise is diminished by reconstructing an estimated signal based partly on measurements and partly on model predictions. This method also tends to suppress recognition of non-noise perturbations due to faults.

A possibly promising approach to noise filtering may exist in the method of scale-space filtering, originally developed for image processing. This method enables the reconstruction of the original signal from a noisy signal by characterizing the signal over a variety of time scales. Witken provides a very informative summary of the method [Wit87, p. 973–980].

5.5.6 Speeding it Up

A practical limitation of the current implementation is that it is slow. QSIM, Q2, and NSIM are research tools that were designed more for clarity and experimentation than for speed. However, as these tools are applied to increasingly large and complex models, the speed of execution begins to become a practical limitation. Three areas for improvement stand out:

1. QSIM's constraint-satisfaction algorithm is presently a simple chronological-backtracking algorithm. At a minimum it should be upgraded to use a dependency-directed backtracking algorithm, a.k.a. backjumping. Also, it could be improved to check for "no-goods", especially if it can be shown to improve speed in the average case.

2. Q2's range-propagation algorithm is slow, but not because the algorithm is naive. Rather, the representation that it manipulates consists of lists of symbols, with numerous associative searches to locate appropriate entries. This propagator can be speeded up significantly by changing to a representation in the form of a network of pointers to appropriate objects that Lisp can manipulate through accessor functions.

3. Much of the code in QSIM and Q2 and MIMIC creates and abandons lists and other objects with no concern about the subsequent price to be paid in garbage collection. Probably the best approach here is to identify the few places that consume the most memory and rewrite them to either avoid consing altogether or at least manage their own memory pools.

5.5.7 Reconciling FDI and MBR

As noted earlier in Chapter 2 in examining Kalman filters, there is a large body of work in the engineering specialty of "fault detection & isolation" (FDI). One thing that is clear from a study of the FDI literature is that model-based approaches to monitoring and diagnosis are not unique to the AI community. The basic concept — that of using a model to predict expected behavior, and then using discrepancies between predictions and observations as diagnostic clues — has evolved in two separate communities: the model-based reasoning (MBR) specialty within the AI community and the fault detection & isolation (FDI) specialty within the engineering community. Unfortunately, the two communities are largely unaware of each other, and each could profit from a better understanding of the other's work. Certainly, the MBR community would profit from an understanding of model-based signal processing and the modern approach to system analysis that it is based on, covering topics such as state estimation, parameter estimation, noise filtering, observability, controllability, and stability. Likewise, the FDI community would benefit from an understanding of qualitative constraint models, constraint suspension, assumption-based truth maintenance, and semiquantitative simulation. An extremely valuable contribution to both communities would be a comprehensive article that compares and contrasts the MBR and FDI methods on a sample problem, showing relative strengths and weaknesses of each.

5.5.8 Real Applications

> *When one takes an FDI scheme whose feasibility has been demonstrated (including a laboratory setting) and attempts to implement that scheme into a practical operating device in a real system, numerous practical and unforeseen difficulties present themselves. To overcome these difficulties one must understand the FDI scheme as well as the nature of the practical problems. This usually requires the FDI designer to follow his work into the specific engineering field, either doing the implementation himself or working closely with those who do it. For this reason we also call* applications *a legitimate area of research.* [PFC89, pages 13–14]

The research presented here has been tested only on relatively simple systems simulated in the laboratory. The next logical step is to test our methods in an application of realistic complexity and scale. As the above quote warns, we can expect to see "numerous practical and unforeseen difficulties", but these will serve to guide future research.

5.6 Epilogue

Given the technologies of semiquantitative simulation and model-based diagnosis as a foundation to build upon, the design of MIMIC is fairly simple and easy to understand. The benefits of the design, however, are almost surprising in extent and importance. In a very real sense, "the whole is greater than the sum of its parts."

It would be dishonest to claim that the MIMIC design is better in every respect than existing methods, and it would be an exaggeration to say that the experimental results presented here "prove" that the design scales up to handle large systems. Rather, we believe that this dissertation has demonstrated a promising approach to monitoring and diagnosis, and that further research driven by real applications will help turn that promise into engineering practice.

Appendix A

Sample Execution History

This appendix shows the unedited execution history of Mimic while monitoring the gravity-flow tank, as described in chapter 4. The main events of interest in this history are summarized below:

- $t = 0$
  The tank is full, `drain` = `normal`, draining begins. Mimic creates the initial qualitative state and Nsim creates the extremal equations that are used in its numerical simulation.

- $t = 1.0001$
  The drain becomes partially clogged, reducing its flow rate by one-third. (This is what occurred, but it isn't visible to Mimic until the next measurement.)

- $t = 1.1$
  A trend discrepancy is detected. Hypotheses are generated for `drain` = `lo-blockage` and `drain` = `hi-blockage`. Both models initialize successfully.

- $t = 1.2$
  A trend discrepancy is detected for the model `drain` = `hi-blockage`, but model `drain` = `lo-blockage` passes all discrepancy tests. Hence, the former is discarded and the latter is retained.

- $t = 1.3$ and beyond.
  Model `drain` = `lo-blockage` continues to be corroborated by subsequent measurements, emerging as the final hypothesis.

- $t = 2.2$
  The very last `PREDICTED` entry shows a somewhat rare case where the lower-bound for `Amount` comes from Q2 and the upper-bound comes from Nsim.

```
Command: (mimic-draining-tank)
initial-qvalues: ((AMOUNT (FULL NIL)) (DRAIN (NORMAL STD)) (TOLERANCE (NORMAL STD)))
Run time: 0.030 seconds to initialize a state.
Initial states: (S-0,.)
Table is
(OUTFLOW (M+ AMOUNT OUTFLOW))
(NETFLOW (- OUTFLOW))
(AMOUNT NETFLOW)
```

```
State equations are
((AMOUNT (- (FCTN (M+ AMOUNT OUTFLOW) AMOUNT))))
Extremal Tables:::
((AMOUNT LB (- (UB (M+ AMOUNT OUTFLOW) (LB AMOUNT))) T))
((AMOUNT UB (- (LB (M+ AMOUNT OUTFLOW) (UB AMOUNT))) T))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
env         = UE1
Creating system function F0121
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'UE1 (SVREF Y 0)))))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
env         = LE1
Creating system function F0122
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'LE1 (SVREF Y 0)))))

--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       0.2            (inc)
READINGS:    Amount     [81.02 86.04]  (dec)      [-80.958 -76.242]
--------------------------------------------------------------------------------

ADV1
EXTENDING:   S-0   to next time point state
Curvatures: NIL
Sd3-constraints:  NIL

SUCCESSORS:  S-0   --> (S-1) [before global filters]
SUCCESSORS:  S-0   --> (S-1) [after  global filters]
SUCCESSORS:  S-1   --> (S-2) [before global filters]
SUCCESSORS:  S-1   --> (S-2) [after  global filters]
TRACKING:    S-0   at time = [0.00 0.00] does not cover time of readings 0.2

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-0,.  --> S-1,-     --> (S-2,.f)

ADV2
INSERTION:   S-1,- ==> (S-3,- S-4,. S-5,-)  [S-4,. will be reading state]
UPDATING:    S-4,.   (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-4,.   AMOUNT     [78.80 88.39]    [81.05 86.07]

------------------------------------------------------------------------------
             State   Variable   Observed         Predicted         Similarity
             -----   ---------  --------------   --------------    ----------
SIMILARITY:  S-4     Time       [0.20 0.20]      [0.20 0.20]       1.000
SIMILARITY:  S-4     Amount     [81.02 86.04]    [81.05 86.07]     0.994
SIMILARITY:  S-4     Amount'    [-80.96 -76.24]  [-101.00 -64.84]  (range of Netflow)
SIMILARITY:  S-4,.   ........................................... = 0.994
------------------------------------------------------------------------------

TRACKED:     S-4,.
```

```
UPDATING:    S-4,.   (updating Q2 values from readings)
UPDATING:    S-4,.   (updating Nsim values from readings)
UPDATING:    S-4     Amount     [81.05 86.04] <-  [81.05 86.07] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = normal
                 states = (S-4)
                 similarity = 0.994   apriori probability = 0.90
                 age = 0.20  [created at 0.00, highest similarity at 0.00]
=============================================================================


-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       0.3            (inc)
READINGS:    Amount     [74.05 78.63]  (dec)      [-74.057 -69.743] [64.990 69.010]
-------------------------------------------------------------------------------

TRACKING:    S-4   --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-4,.  --> S-5,-     --> (S-2,.f)

ADV3
INSERTION:   S-5,- ==> (S-6,- S-7,. S-8,-)  [S-7,. will be reading state]
UPDATING:    S-7,.   (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                Nsim
PREDICTED:   S-7,.   AMOUNT     [72.45 80.24]     [73.34 79.42]

-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
SIMILARITY:  S-7     Time       [0.30 0.30]       [0.30 0.30]       1.000
SIMILARITY:  S-7     Amount     [74.05 78.63]     [73.34 79.42]     0.992
SIMILARITY:  S-7     Amount'    [-74.06 -69.74]   [-86.04 -58.67]   (range of Netflow)
SIMILARITY:  S-7     Amount''   [64.990 69.010]   inc               (qdir  of Netflow)
SIMILARITY:  S-7,.   ......................................... = 0.992
-----------------------------------------------------------------------------

TRACKED:     S-7,.
UPDATING:    S-7,.   (updating Q2 values from readings)
UPDATING:    S-7,.   (updating Nsim values from readings)
UPDATING:    S-7     Amount     [74.05 78.63] <-  [73.34 79.42] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = normal
                 states = (S-7)
                 similarity = 0.992   apriori probability = 0.90
                 age = 0.30  [created at 0.00, highest similarity at 0.00]
=============================================================================


-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
```

```
READINGS:    Time       0.8            (inc)
READINGS:    Amount     [47.21 50.13]  (dec)      [-57.000 -53.680] [53.544 56.856]
--------------------------------------------------------------------------------

TRACKING:    S-7   --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-7,.  --> S-8,-     --> (S-2,.f)

ADV3
INSERTION:   S-8,- ==> (S-9,- S-10,. S-11,-)  [S-10,. will be reading state]
UPDATING:    S-10,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-10,.  AMOUNT     [34.73 64.74]    [44.91 52.71]

------------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-10    Time       [0.80 0.80]      [0.80 0.80]      1.000
SIMILARITY:  S-10    Amount     [47.21 50.13]    [44.91 52.71]    0.974
SIMILARITY:  S-10    Amount'    [-57.00 -53.68]  [-78.63 -35.93]  (range of Netflow)
SIMILARITY:  S-10    Amount''   [53.544 56.856]  inc              (qdir  of Netflow)
SIMILARITY:  S-10,.  ......................................... = 0.974
------------------------------------------------------------------------------

TRACKED:     S-10,.
UPDATING:    S-10,.  (updating Q2 values from readings)
UPDATING:    S-10,.  (updating Nsim values from readings)
UPDATING:    S-10    Amount     [47.21 50.13] <-  [44.91 52.71] (updating nsim)

==============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = normal
                 states = (S-10)
                 similarity = 0.974   apriori probability = 0.90
                 age = 0.80  [created at 0.00, highest similarity at 0.00]
==============================================================================

--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       0.9            (inc)
READINGS:    Amount     [43.15 45.81]  (dec)      [-43.157 -40.643] [43.456 46.144]
--------------------------------------------------------------------------------

TRACKING:    S-10  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-10,. --> S-11,-    --> (S-2,.f)

ADV3
INSERTION:   S-11,- ==> (S-12,- S-13,. S-14,-)  [S-13,. will be reading state]
UPDATING:    S-13,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-13,.  AMOUNT     [42.20 46.75]    [42.72 46.28]

------------------------------------------------------------------------------
```

```
            State   Variable   Observed         Predicted        Similarity
            -----   ---------  --------------   --------------   ----------
SIMILARITY: S-13    Time       [0.90 0.90]      [0.90 0.90]      1.000
SIMILARITY: S-13    Amount     [43.15 45.81]    [42.72 46.28]    0.995
SIMILARITY: S-13    Amount'    [-43.16 -40.64]  [-50.13 -34.17]  (range of Netflow)
SIMILARITY: S-13    Amount''   [43.456 46.144]  inc              (qdir  of Netflow)
SIMILARITY: S-13,.  ........................................... = 0.995
-----------------------------------------------------------------------------

TRACKED:    S-13,.
UPDATING:   S-13,.  (updating Q2 values from readings)
UPDATING:   S-13,.  (updating Nsim values from readings)
UPDATING:   S-13    Amount     [43.15 45.81] <-  [42.72 46.28] (updating nsim)

=============================================================================
CANDIDATES: 1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

            1:  Drain = normal
                states = (S-13)
                similarity = 0.995   apriori probability = 0.90
                age = 0.90  [created at 0.00, highest similarity at 0.00]
=============================================================================

-------------------------------------------------------------------------------
            VARIABLE   VALUE         QDIRS      1st-DERIV      2nd-DERIV
            ---------  ------------- ---------  -------------  ---------------
READINGS:   Time       1.0           (inc)
READINGS:   Amount     [39.44 41.88]  (dec)      [-39.346 -37.054] [35.890 38.110]
-------------------------------------------------------------------------------

TRACKING:   S-13  --> already tracked; moving to its successors

SUCCESSORS: Point ---> Interval ---> Point
SUCCESSORS: S-13,. --> S-14,-    --> (S-2,.f)

ADV3
INSERTION:  S-14,- ==> (S-15,- S-16,. S-17,-)  [S-16,. will be reading state]
UPDATING:   S-16,.  (updating Q2 values from Nsim predictions)

PREDICTED:  State   Varname    Q2               Nsim
PREDICTED:  S-16,.  AMOUNT     [38.56 42.73]    [39.04 42.29]

-----------------------------------------------------------------------------
            State   Variable   Observed         Predicted        Similarity
            -----   ---------  --------------   --------------   ----------
SIMILARITY: S-16    Time       [1.00 1.00]      [1.00 1.00]      1.000
SIMILARITY: S-16    Amount     [39.44 41.88]    [39.04 42.29]    0.998
SIMILARITY: S-16    Amount'    [-39.35 -37.05]  [-45.81 -31.23]  (range of Netflow)
SIMILARITY: S-16    Amount''   [35.890 38.110]  inc              (qdir  of Netflow)
SIMILARITY: S-16,.  ........................................... = 0.998
-----------------------------------------------------------------------------

TRACKED:    S-16,.
UPDATING:   S-16,.  (updating Q2 values from readings)
UPDATING:   S-16,.  (updating Nsim values from readings)
UPDATING:   S-16    Amount     [39.44 41.88] <-  [39.04 42.29] (updating nsim)

=============================================================================
CANDIDATES: 1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.
```

```
            1:  Drain = normal
                states = (S-16)
                similarity = 0.998   apriori probability = 0.90
                age = 1.00  [created at 0.00, highest similarity at 0.00]
==============================================================================

--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.1            (inc)
READINGS:    Amount     [37.14 39.44]  (dec)      [-24.411 -22.989] [140.650 149.350]
--------------------------------------------------------------------------------

TRACKING:    S-16  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-16,. --> S-17,-    --> (S-2,.f)

ADV3
INSERTION:   S-17,- ==> (S-18,- S-19,. S-20,-)  [S-19,. will be reading state]
UPDATING:    S-19,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-19,.  AMOUNT     [35.25 39.06]    [35.69 38.66]

------------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-19    Time       [1.10 1.10]      [1.10 1.10]      1.000
SIMILARITY:  S-19    Amount     [37.14 39.44]    [35.69 38.66]    0.576
SIMILARITY:  S-19    Amount'    [-24.41 -22.99]  [-41.88 -28.55]  (range of Netflow) <--
SIMILARITY:  S-19    Amount''   [140.650 149.350] inc             (qdir  of Netflow)
SIMILARITY:  S-19,.  ........................................ = RATE
------------------------------------------------------------------------------

TRACKING:    S-19,. --> Rate discrepancy, is being discarded.
DISCREP:     S-19,. discrepancies (AMOUNT) saved on S-17,-
REMOVAL:     (S-18,- S-19,. S-20,-) ==> S-17,-  [restoring S-17,-]

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:             S-17,-    --> S-2,.f

ADV2
EXTENDING:   S-2   is a quiescent/final state; cannot be extended.
INSERTION:   S-2,.f ==> S-21,.f  [S-21,.f will be reading state]
UPDATING:    S-21,.f (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-21,.# AMOUNT     [19.00 20.00]    [35.69 38.66]
UPDATING:    S-21,.# discrepancies = (TIME)
TRACKING:    S-21,.# being discarded because of inconsistency with Nsim values.
DISCREP:     S-21,.# discrepancies (TIME) saved on S-2,.f
REMOVAL:     S-21,.# ==> S-2,.f  [restoring S-2,.f]
SUCCESSORS:  S-2,.f --> has no successors; status = (TRANSITION FINAL COMPLETE)
TRACING:     USE-DISCREPANCIES from  S-17,-: (AMOUNT)
TRACING:     Amount evokes suspects (Drain tank tolerance amount-sensor)
TRACING:     Final suspects = (Drain tank tolerance amount-sensor)
HYPOTHESES:  Removing suspect TANK; it's not a mode variable
HYPOTHESES:  Removing suspect AMOUNT-SENSOR; it's not a mode variable
TRACING:             Tolerance is not a known mode variable.
```

```
---------------------------------------------------------------------------
HYPOTHESES:  Single-change hypotheses:
             1:  Drain = lo-blockage  (prob = 0.0700)
             2:  Drain = hi-blockage  (prob = 0.0300)
---------------------------------------------------------------------------
---------------------------------------------------------------------------
HYPOTHESES:  New (not old) hypotheses:
             1:  Drain = lo-blockage  (prob = 0.0700)
             2:  Drain = hi-blockage  (prob = 0.0300)
---------------------------------------------------------------------------

----------------------------------------------------------------------------
INITIALIZE:  Creating new state(s) from S-16 + hypothesized modes + readings.
INITIALIZE:  Drain = hi-blockage

             Var type     Variable     Qval           Reading
             -----------  ----------   ------------   -------------------
             time         Time         (t*-6 inc)     (1.1 1.1)
             reading      Amount       (a-0 dec)      (37.1413 39.4387)
             dependent    Outflow      (NIL nil)
             dependent    Netflow      (NIL nil)
             dependent    Amount-obs   (NIL nil)
             independent  Tolerance    (normal std)
             mode         Drain        (hi-blockage std)

Run time: 0.025 seconds to initialize a state.
INITIALIZE:  S-22,. --completions--> (S-22,.)
ZIP-UP (TIME t*-6):  (-INF +INF) -> (0.9999999 +INF).
Updating:  (TIME t*-6) from (0.9999999 +INF) -> (1.0999999 1.1000001).
Insignificant update ignored (TOLERANCE normal):  (0.9799999 1.0200001) ~ (0.98 1.02).
Updating:  (OUTFLOW (AT S-22,.)) from (-INF +INF) -> (-INF 101).
Updating:  (OUTFLOW (AT S-22,.)) from (-INF 101) -> (0 101).
Updating:  (NETFLOW (AT S-22,.)) from (-INF +INF) -> (-101 0).
Updating:  (AMOUNT-OBS (AT S-22,.)) from (-INF +INF) -> (0 +INF).
Updating:  (OUTFLOW (AT S-22,.)) from (0 101) -> (0 17.06368).
Updating:  (NETFLOW (AT S-22,.)) from (-101 0) -> (-17.06368 0).
Updating:  (AMOUNT-OBS (AT S-22,.)) from (0 +INF) -> (37.792847 43.512386).
ZIP-UP (TIME t1):  (0.9999999 +INF) -> (1.0999999 +INF).
Q2 assertions = (((TIME t*-6) (1.1 1.1)) ((AMOUNT a-0) (37.1413 39.4387)))
Table is
(OUTFLOW (M+ AMOUNT OUTFLOW))
(NETFLOW (- OUTFLOW))
(AMOUNT NETFLOW)
State equations are
((AMOUNT (- (FCTN (M+ AMOUNT OUTFLOW) AMOUNT))))
Extremal Tables:::
((AMOUNT LB (- (UB (M+ AMOUNT OUTFLOW) (LB AMOUNT))) T))
((AMOUNT UB (- (LB (M+ AMOUNT OUTFLOW) (UB AMOUNT))) T))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
env         = UE1
Creating system function F0123
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'UE1 (SVREF Y 0)))))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
```

```
env        = LE1
Creating system function F0124
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'LE1 (SVREF Y 0)))))
INITIALIZE:         ---filtered----> (S-22,.)
---------------------------------------------------------------------------
UNDO:       Created S-23,. from S-22,. by removing lmarks created for S-16,.

------------------------------------------------------------------------
RESIMULATE:  reading sequence   = (S-4 S-7 S-10 S-13 S-16 S-22)

RESIMULATE:  resim sub-sequence = (S-16 S-22)
RESIMULATE:  starting from S-16 (time = 1.0)
RESIMULATE:  intersecting with readings from S-22 (time = 1.1)
UPDATING:    S-25    (updating Q2 values from readings)
UPDATING:    S-25    (updating Nsim values from readings)
UPDATING:    S-25    Amount     [37.89 39.44] <-  [37.89 41.88] (updating nsim)
RESIMULATE:  S-25    sim = 0.49175772

RESIMULATE:  resim sub-sequence = (S-13 S-16 S-22)
RESIMULATE:  starting from S-13 (time = 0.9)
RESIMULATE:  intersecting with readings from S-16 (time = 1.0)
UPDATING:    S-27    (updating Q2 values from readings)
UPDATING:    S-27    (updating Nsim values from readings)
UPDATING:    S-27    Amount     [41.45 41.88] <-  [41.45 45.81] (updating nsim)
RESIMULATE:  intersecting with readings from S-22 (time = 1.1)
UPDATING:    S-29    (updating Q2 values from readings)
UPDATING:    S-29    (updating Nsim values from readings)
UPDATING:    S-29    Amount     [39.83 39.44] <-  [39.83 41.88] (updating nsim)
                     discarding because lower bound exceeds upper bound!
RESIMULATE:  S-29    sim = 0

RESIMULATE:  best-state = S-25, best-sim = 0.49175772, best-time = 1.0
------------------------------------------------------------------------

EXTENDING:   S-25  to next time point state
SUCCESSORS:  S-25  --> (S-30) [before global filters]
SUCCESSORS:  S-25  --> (S-30) [after  global filters]
SUCCESSORS:  S-30  --> (S-31) [before global filters]
SUCCESSORS:  S-30  --> (S-31) [after  global filters]
HYPOTHESES:  (Drain = hi-blockage) ---> (S-25,.)


---------------------------------------------------------------------------
INITIALIZE:  Creating new state(s) from S-16 + hypothesized modes + readings.
INITIALIZE:  Drain = lo-blockage

             Var type     Variable     Qval           Reading
             -----------  ----------   ------------   -------------------
             time         Time         (t*-10 inc)    (1.1 1.1)
             reading      Amount       (a-0 dec)      (37.1413 39.4387)
             dependent    Outflow      (NIL nil)
             dependent    Netflow      (NIL nil)
             dependent    Amount-obs   (NIL nil)
             independent  Tolerance    (normal std)
             mode         Drain        (lo-blockage std)

Run time: 0.042 seconds to initialize a state.
INITIALIZE:  S-32,. --completions--> (S-32,.)
ZIP-UP (TIME t*-10):  (-INF +INF) -> (0.9999999 +INF).
```

```
Updating:  (TIME t*-10) from (0.9999999 +INF) -> (1.0999999 1.1000001).
Insignificant update ignored (TOLERANCE normal):  (0.9799999 1.0200001) ~ (0.98 1.02).
Updating:  (OUTFLOW (AT S-32,.)) from (-INF +INF) -> (-INF 101).
Updating:  (OUTFLOW (AT S-32,.)) from (-INF 101) -> (0 101).
Updating:  (NETFLOW (AT S-32,.)) from (-INF +INF) -> (-101 0).
Updating:  (AMOUNT-OBS (AT S-32,.)) from (-INF +INF) -> (0 +INF).
Updating:  (OUTFLOW (AT S-32,.)) from (0 101) -> (15.425653 34.12736).
Updating:  (NETFLOW (AT S-32,.)) from (-101 0) -> (-34.12736 -15.425653).
Updating:  (AMOUNT-OBS (AT S-32,.)) from (0 +INF) -> (37.792847 43.512386).
ZIP-UP (TIME t1):  (0.9999999 +INF) -> (1.0999999 +INF).
Q2 assertions = (((TIME t*-10) (1.1 1.1)) ((AMOUNT a-0) (37.1413 39.4387)))
Table is
(OUTFLOW (M+ AMOUNT OUTFLOW))
(NETFLOW (- OUTFLOW))
(AMOUNT NETFLOW)
State equations are
((AMOUNT (- (FCTN (M+ AMOUNT OUTFLOW) AMOUNT))))
Extremal Tables:::
((AMOUNT LB (- (UB (M+ AMOUNT OUTFLOW) (LB AMOUNT))) T))
((AMOUNT UB (- (LB (M+ AMOUNT OUTFLOW) (UB AMOUNT))) T))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
env         = UE1
Creating system function F0125
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'UE1 (SVREF Y 0)))))
M-envelopes = (((M+ AMOUNT OUTFLOW) (UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
                                    (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1)))
env-set     = ((UPPER-ENVELOPE UE1) (UPPER-INVERSE UI1)
               (LOWER-ENVELOPE LE1) (LOWER-INVERSE LI1))
env         = LE1
Creating system function F0126
(LAMBDA (TIN Y YP)
  (DECLARE (IGNORE TIN)) (SETF (SVREF YP 0) (- (FUNCALL #'LE1 (SVREF Y 0)))))
INITIALIZE:          ---filtered----> (S-32,.)
----------------------------------------------------------------------------
UNDO:        Created S-33,. from S-32,. by removing lmarks created for S-16,.

-------------------------------------------------------------------------
RESIMULATE:  reading sequence   = (S-4 S-7 S-10 S-13 S-16 S-32)

RESIMULATE:  resim sub-sequence = (S-16 S-32)
RESIMULATE:  starting from S-16 (time = 1.0)
RESIMULATE:  intersecting with readings from S-32 (time = 1.1)
UPDATING:    S-35    (updating Q2 values from readings)
UPDATING:    S-35    (updating Nsim values from readings)
UPDATING:    S-35    Amount     [37.14 39.44] <-  [36.41 40.24] (updating nsim)
RESIMULATE:  S-35    sim = 0.9893015

RESIMULATE:  resim sub-sequence = (S-13 S-16 S-32)
RESIMULATE:  starting from S-13 (time = 0.9)
RESIMULATE:  intersecting with readings from S-16 (time = 1.0)
UPDATING:    S-37    (updating Q2 values from readings)
UPDATING:    S-37    (updating Nsim values from readings)
UPDATING:    S-37    Amount     [39.83 41.88] <-  [39.83 44.02] (updating nsim)
RESIMULATE:  intersecting with readings from S-32 (time = 1.1)
UPDATING:    S-39    (updating Q2 values from readings)
UPDATING:    S-39    (updating Nsim values from readings)
```

```
UPDATING:    S-39    Amount     [37.14 39.44] <-  [36.77 40.24] (updating nsim)
RESIMULATE:  S-39    sim = 0.92651534

RESIMULATE:  best-state = S-35, best-sim = 0.9893015, best-time = 1.0
------------------------------------------------------------------------

EXTENDING:   S-35  to next time point state
SUCCESSORS:  S-35  --> (S-40) [before global filters]
SUCCESSORS:  S-35  --> (S-40) [after  global filters]
SUCCESSORS:  S-40  --> (S-41) [before global filters]
SUCCESSORS:  S-40  --> (S-41) [after  global filters]
HYPOTHESES:  (Drain = lo-blockage) ---> (S-35,.)

=============================================================================
CANDIDATES:  1 original, 0 retained, 1 rejected, 2 hypothesized, 2 new.

             1:  Drain = hi-blockage
                 states = (S-25)
                 similarity = 0.492   apriori probability = 0.03
                 age = 0.10  [created at 1.10, highest similarity at 1.00]

             2:  Drain = lo-blockage
                 states = (S-35)
                 similarity = 0.989   apriori probability = 0.07
                 age = 0.10  [created at 1.10, highest similarity at 1.00]
=============================================================================

-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.2            (inc)
READINGS:    Amount     [34.98 37.14]  (dec)      [-22.969 -21.631] [13.580 14.420]
-------------------------------------------------------------------------------

ADV3
EXTENDING:   S-25  to next time point state
TRACKING:    S-25  at time = [1.10 1.10] does not cover time of readings 1.2

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-25,. --> S-30,-    --> (S-31,.f)

ADV2
INSERTION:   S-30,- ==> (S-42,- S-43,. S-44,-)  [S-43,. will be reading state]
UPDATING:    S-43,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                Nsim
PREDICTED:   S-43,.  AMOUNT     [35.56 39.44]     [36.41 39.44]

-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
SIMILARITY:  S-43    Time       [1.20 1.20]       [1.20 1.20]       1.000
SIMILARITY:  S-43    Amount     [34.98 37.14]     [36.41 39.44]     0.283
SIMILARITY:  S-43    Amount'    [-22.97 -21.63]   [-15.78 0.00]     (range of Netflow) <--
SIMILARITY:  S-43    Amount''   [13.580 14.420]   inc               (qdir  of Netflow)
SIMILARITY:  S-43,.  .......................................... = RATE
-----------------------------------------------------------------------------

TRACKING:    S-43,. --> Rate discrepancy, is being discarded.
DISCREP:     S-43,. discrepancies (AMOUNT) saved on S-30,-
```

```
REMOVAL:     (S-42,- S-43,. S-44,-) ==> S-30,-  [restoring S-30,-]

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:             S-30,-    --> S-31,.f

ADV2
EXTENDING:   S-31  is a quiescent/final state; cannot be extended.
INSERTION:   S-31,.f ==> S-45,.f  [S-45,.f will be reading state]
UPDATING:    S-45,.f (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2              Nsim
PREDICTED:   S-45,.# AMOUNT     [19.00 20.00]   [36.41 39.44]
UPDATING:    S-45,.# discrepancies = (TIME)
TRACKING:    S-45,.# being discarded because of inconsistency with Nsim values.
DISCREP:     S-45,.# discrepancies (TIME) saved on S-31,.f
REMOVAL:     S-45,.# ==> S-31,.f  [restoring S-31,.f]
SUCCESSORS:  S-31,.f --> has no successors; status = (TRANSITION FINAL COMPLETE)
ADV3
EXTENDING:   S-35  to next time point state
TRACKING:    S-35  at time = [1.10 1.10] does not cover time of readings 1.2

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-35,. --> S-40,-    --> (S-41,.f)

ADV2
INSERTION:   S-40,- ==> (S-46,- S-47,. S-48,-)  [S-47,. will be reading state]
UPDATING:    S-47,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2              Nsim
PREDICTED:   S-47,.  AMOUNT     [33.99 38.08]   [34.29 37.89]

------------------------------------------------------------------------------
             State   Variable   Observed         Predicted         Similarity
             -----   ---------  --------------   --------------    ----------
SIMILARITY:  S-47    Time       [1.20 1.20]      [1.20 1.20]       1.000
SIMILARITY:  S-47    Amount     [34.98 37.14]    [34.29 37.89]     0.990
SIMILARITY:  S-47    Amount'    [-22.97 -21.63]  [-31.55 -13.71]   (range of Netflow)
SIMILARITY:  S-47    Amount''   [13.580 14.420]  inc               (qdir  of Netflow)
SIMILARITY:  S-47,.  ........................................... = 0.990
------------------------------------------------------------------------------

TRACKED:     S-47,.
UPDATING:    S-47,.  (updating Q2 values from readings)
UPDATING:    S-47,.  (updating Nsim values from readings)
UPDATING:    S-47    Amount     [34.98 37.14] <-  [34.29 37.89] (updating nsim)

==============================================================================
CANDIDATES:  2 original, 1 retained, 1 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-47)
                 similarity = 0.990   apriori probability = 0.07
                 age = 0.20  [created at 1.10, highest similarity at 1.00]
==============================================================================

--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.3            (inc)
READINGS:    Amount     [32.94 34.98]  (dec)      [-21.630 -20.370] [12.609 13.390]
```

```
-------------------------------------------------------------------------------

TRACKING:    S-47  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-47,. --> S-48,-    --> (S-41,.f)

ADV3
INSERTION:   S-48,- ==> (S-49,- S-50,. S-51,-)  [S-50,. will be reading state]
UPDATING:    S-50,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                Nsim
PREDICTED:   S-50,.  AMOUNT     [32.01 35.86]     [32.29 35.69]

-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
SIMILARITY:  S-50    Time       [1.30 1.30]       [1.30 1.30]       1.000
SIMILARITY:  S-50    Amount     [32.94 34.98]     [32.29 35.69]     0.990
SIMILARITY:  S-50    Amount'    [-21.63 -20.37]   [-29.71 -12.92]   (range of Netflow)
SIMILARITY:  S-50    Amount''   [12.609 13.390]   inc               (qdir  of Netflow)
SIMILARITY:  S-50,.  ........................................... = 0.990
-----------------------------------------------------------------------------

TRACKED:     S-50,.
UPDATING:    S-50,.  (updating Q2 values from readings)
UPDATING:    S-50,.  (updating Nsim values from readings)
UPDATING:    S-50    Amount     [32.94 34.98] <-  [32.29 35.69] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-50)
                 similarity = 0.990   apriori probability = 0.07
                 age = 0.30  [created at 1.10, highest similarity at 1.00]
=============================================================================

-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.4            (inc)
READINGS:    Amount     [31.02 32.94]  (dec)      [-20.394 -19.206] [11.640 12.361]
-------------------------------------------------------------------------------

TRACKING:    S-50  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-50,. --> S-51,-    --> (S-41,.f)

ADV3
INSERTION:   S-51,- ==> (S-52,- S-53,. S-54,-)  [S-53,. will be reading state]
UPDATING:    S-53,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                Nsim
PREDICTED:   S-53,.  AMOUNT     [30.14 33.77]     [30.41 33.61]

-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
```

```
SIMILARITY: S-53    Time       [1.40 1.40]       [1.40 1.40]       1.000
SIMILARITY: S-53    Amount     [31.02 32.94]     [30.41 33.61]     0.989
SIMILARITY: S-53    Amount'    [-20.39 -19.21]   [-27.98 -12.16]   (range of Netflow)
SIMILARITY: S-53    Amount''   [11.640 12.361]   inc               (qdir  of Netflow)
SIMILARITY: S-53,.  ............................................ = 0.989
----------------------------------------------------------------------------

TRACKED:    S-53,.
UPDATING:   S-53,.  (updating Q2 values from readings)
UPDATING:   S-53,.  (updating Nsim values from readings)
UPDATING:   S-53    Amount     [31.02 32.94] <-  [30.41 33.61] (updating nsim)

============================================================================
CANDIDATES: 1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

            1:  Drain = lo-blockage
                states = (S-53)
                similarity = 0.989   apriori probability = 0.07
                age = 0.40  [created at 1.10, highest similarity at 1.00]
============================================================================

------------------------------------------------------------------------------
            VARIABLE   VALUE         QDIRS      1st-DERIV      2nd-DERIV
            ---------  ------------- ---------  -------------  ---------------
READINGS:   Time       1.5           (inc)
READINGS:   Amount     [29.22 31.02] (dec)      [-19.158 -18.042] [11.640 12.360]
------------------------------------------------------------------------------

TRACKING:   S-53  --> already tracked; moving to its successors

SUCCESSORS: Point ---> Interval ---> Point
SUCCESSORS: S-53,. --> S-54,-    --> (S-41,.f)

ADV3
INSERTION:  S-54,- ==> (S-55,- S-56,. S-57,-)  [S-56,. will be reading state]
UPDATING:   S-56,.  (updating Q2 values from Nsim predictions)

PREDICTED:  State   Varname    Q2                Nsim
PREDICTED:  S-56,.  AMOUNT     [28.39 31.80]     [28.64 31.65]

----------------------------------------------------------------------------
            State   Variable   Observed          Predicted         Similarity
            -----   ---------  --------------    --------------    ----------
SIMILARITY: S-56    Time       [1.50 1.50]       [1.50 1.50]       1.000
SIMILARITY: S-56    Amount     [29.22 31.02]     [28.64 31.65]     0.991
SIMILARITY: S-56    Amount'    [-19.16 -18.04]   [-26.35 -11.45]   (range of Netflow)
SIMILARITY: S-56    Amount''   [11.640 12.360]   inc               (qdir  of Netflow)
SIMILARITY: S-56,.  ............................................ = 0.991
----------------------------------------------------------------------------

TRACKED:    S-56,.
UPDATING:   S-56,.  (updating Q2 values from readings)
UPDATING:   S-56,.  (updating Nsim values from readings)
UPDATING:   S-56    Amount     [29.22 31.02] <-  [28.64 31.65] (updating nsim)

============================================================================
CANDIDATES: 1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

            1:  Drain = lo-blockage
                states = (S-56)
```

```
                 similarity = 0.991   apriori probability = 0.07
                 age = 0.50  [created at 1.10, highest similarity at 1.00]
==============================================================================


--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.6            (inc)
READINGS:    Amount     [27.51 29.21]  (dec)      [-18.128 -17.072] [9.700 10.300]
--------------------------------------------------------------------------------

TRACKING:    S-56  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-56,. --> S-57,-    --> (S-41,.f)

ADV3
INSERTION:   S-57,- ==> (S-58,- S-59,. S-60,-)  [S-59,. will be reading state]
UPDATING:    S-59,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                 Nsim
PREDICTED:   S-59,.  AMOUNT     [26.73 29.95]      [26.97 29.81]

------------------------------------------------------------------------------
             State   Variable   Observed           Predicted          Similarity
             -----   ---------  --------------     --------------     ----------
SIMILARITY:  S-59    Time       [1.60 1.60]        [1.60 1.60]        1.000
SIMILARITY:  S-59    Amount     [27.51 29.21]      [26.97 29.81]      0.987
SIMILARITY:  S-59    Amount'    [-18.13 -17.07]    [-24.82 -10.79]    (range of Netflow)
SIMILARITY:  S-59    Amount''   [9.700 10.300]     inc                (qdir  of Netflow)
SIMILARITY:  S-59,.  ........................................... = 0.987
------------------------------------------------------------------------------

TRACKED:     S-59,.
UPDATING:    S-59,.  (updating Q2 values from readings)
UPDATING:    S-59,.  (updating Nsim values from readings)
UPDATING:    S-59    Amount     [27.51 29.21] <-  [26.97 29.81] (updating nsim)

==============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-59)
                 similarity = 0.987   apriori probability = 0.07
                 age = 0.60  [created at 1.10, highest similarity at 1.00]
==============================================================================


--------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.7            (inc)
READINGS:    Amount     [25.91 27.51]  (dec)      [-16.995 -16.005] [10.670 11.330]
--------------------------------------------------------------------------------

TRACKING:    S-59  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-59,. --> S-60,-    --> (S-41,.f)
```

```
ADV3
INSERTION:   S-60,- ==> (S-61,- S-62,. S-63,-)  [S-62,. will be reading state]
UPDATING:    S-62,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-62,.  AMOUNT     [25.17 28.20]    [25.39 28.07]

-------------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-62    Time       [1.70 1.70]      [1.70 1.70]      1.000
SIMILARITY:  S-62    Amount     [25.91 27.51]    [25.39 28.07]    0.991
SIMILARITY:  S-62    Amount'    [-17.00 -16.01]  [-23.37 -10.16]  (range of Netflow)
SIMILARITY:  S-62    Amount''   [10.670 11.330]  inc              (qdir  of Netflow)
SIMILARITY:  S-62,.  ........................................... = 0.991
-------------------------------------------------------------------------------

TRACKED:     S-62,.
UPDATING:    S-62,.  (updating Q2 values from readings)
UPDATING:    S-62,.  (updating Nsim values from readings)
UPDATING:    S-62    Amount     [25.91 27.51] <-  [25.39 28.07] (updating nsim)

===============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-62)
                 similarity = 0.991   apriori probability = 0.07
                 age = 0.70  [created at 1.10, highest similarity at 1.00]
===============================================================================

---------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.8            (inc)
READINGS:    Amount     [24.41 25.91]  (dec)      [-15.965 -15.035] [9.700 10.300]
---------------------------------------------------------------------------------

TRACKING:    S-62  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-62,. --> S-63,-    --> (S-41,.f)

ADV3
INSERTION:   S-63,- ==> (S-64,- S-65,. S-66,-)  [S-65,. will be reading state]
UPDATING:    S-65,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-65,.  AMOUNT     [23.71 26.56]    [23.92 26.43]

-------------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-65    Time       [1.80 1.80]      [1.80 1.80]      1.000
SIMILARITY:  S-65    Amount     [24.41 25.91]    [23.92 26.43]    0.993
SIMILARITY:  S-65    Amount'    [-15.97 -15.04]  [-22.01 -9.57]   (range of Netflow)
SIMILARITY:  S-65    Amount''   [9.700 10.300]   inc              (qdir  of Netflow)
SIMILARITY:  S-65,.  ........................................... = 0.993
-------------------------------------------------------------------------------
```

```
TRACKED:     S-65,.
UPDATING:    S-65,.  (updating Q2 values from readings)
UPDATING:    S-65,.  (updating Nsim values from readings)
UPDATING:    S-65    Amount     [24.41 25.91] <-  [23.92 26.43] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-65)
                 similarity = 0.993   apriori probability = 0.07
                 age = 0.80  [created at 1.10, highest similarity at 1.00]
=============================================================================


-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       1.9            (inc)
READINGS:    Amount     [22.98 24.40]  (dec)      [-15.141 -14.259] [7.760 8.240]
-------------------------------------------------------------------------------

TRACKING:    S-65  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-65,. --> S-66,-    --> (S-41,.f)

ADV3
INSERTION:   S-66,- ==> (S-67,- S-68,. S-69,-)  [S-68,. will be reading state]
UPDATING:    S-68,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2              Nsim
PREDICTED:   S-68,.  AMOUNT     [22.33 25.02]   [22.53 24.90]

-----------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-68    Time       [1.90 1.90]      [1.90 1.90]      1.000
SIMILARITY:  S-68    Amount     [22.98 24.40]    [22.53 24.90]    0.987
SIMILARITY:  S-68    Amount'    [-15.14 -14.26]  [-20.73 -9.01]   (range of Netflow)
SIMILARITY:  S-68    Amount''   [7.760 8.240]    inc              (qdir  of Netflow)
SIMILARITY:  S-68,.  ......................................... = 0.987
-----------------------------------------------------------------------------

TRACKED:     S-68,.
UPDATING:    S-68,.  (updating Q2 values from readings)
UPDATING:    S-68,.  (updating Nsim values from readings)
UPDATING:    S-68    Amount     [22.98 24.40] <-  [22.53 24.90] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-68)
                 similarity = 0.987   apriori probability = 0.07
                 age = 0.90  [created at 1.10, highest similarity at 1.00]
=============================================================================


-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
```

```
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       2.0            (inc)
READINGS:    Amount     [21.64 22.98]  (dec)      [-14.214 -13.386] [8.730 9.270]
-------------------------------------------------------------------------------

TRACKING:    S-68  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-68,. --> S-69,-    --> (S-41,.f)

ADV3
INSERTION:   S-69,- ==> (S-70,- S-71,. S-72,-)  [S-71,. will be reading state]
UPDATING:    S-71,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-71,.  AMOUNT     [21.03 23.56]    [21.21 23.44]

-----------------------------------------------------------------------------
             State   Variable   Observed         Predicted        Similarity
             -----   ---------  --------------   --------------   ----------
SIMILARITY:  S-71    Time       [2.00 2.00]      [2.00 2.00]      1.000
SIMILARITY:  S-71    Amount     [21.64 22.98]    [21.21 23.44]    0.990
SIMILARITY:  S-71    Amount'    [-14.21 -13.39]  [-19.52 -8.49]   (range of Netflow)
SIMILARITY:  S-71    Amount''   [8.730 9.270]    inc              (qdir  of Netflow)
SIMILARITY:  S-71,.  ......................................... = 0.990
-----------------------------------------------------------------------------

TRACKED:     S-71,.
UPDATING:    S-71,.  (updating Q2 values from readings)
UPDATING:    S-71,.  (updating Nsim values from readings)
UPDATING:    S-71    Amount     [21.64 22.98] <-  [21.21 23.44] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-71)
                 similarity = 0.990   apriori probability = 0.07
                 age = 1.00  [created at 1.10, highest similarity at 1.00]
=============================================================================

-------------------------------------------------------------------------------
             VARIABLE   VALUE          QDIRS      1st-DERIV      2nd-DERIV
             ---------  -------------  ---------  -------------  ---------------
READINGS:    Time       2.1            (inc)
READINGS:    Amount     [20.38 21.64]  (dec)      [-13.390 -12.610] [7.760 8.240]
-------------------------------------------------------------------------------

TRACKING:    S-71  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-71,. --> S-72,-    --> (S-41,.f)

ADV3
INSERTION:   S-72,- ==> (S-73,- S-74,. S-75,-)  [S-74,. will be reading state]
UPDATING:    S-74,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2               Nsim
PREDICTED:   S-74,.  AMOUNT     [19.80 22.19]    [19.98 22.08]
```

```
-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
SIMILARITY:  S-74    Time       [2.10 2.10]       [2.10 2.10]       1.000
SIMILARITY:  S-74    Amount     [20.38 21.64]     [19.98 22.08]     0.990
SIMILARITY:  S-74    Amount'    [-13.39 -12.61]   [-18.38 -7.99]    (range of Netflow)
SIMILARITY:  S-74    Amount''   [7.760 8.240]     inc               (qdir  of Netflow)
SIMILARITY:  S-74,.  ........................................... = 0.990
-----------------------------------------------------------------------------

TRACKED:     S-74,.
UPDATING:    S-74,.  (updating Q2 values from readings)
UPDATING:    S-74,.  (updating Nsim values from readings)
UPDATING:    S-74    Amount     [20.38 21.64] <-  [19.98 22.08] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.

             1:  Drain = lo-blockage
                 states = (S-74)
                 similarity = 0.990   apriori probability = 0.07
                 age = 1.10  [created at 1.10, highest similarity at 1.00]
=============================================================================

-------------------------------------------------------------------------------
             VARIABLE   VALUE         QDIRS      1st-DERIV      2nd-DERIV
             ---------  ------------- ---------  -------------  ---------------
READINGS:    Time       2.2           (inc)
READINGS:    Amount     [19.19 20.37] (dec)      [-12.669 -11.931] [6.790 7.210]
-------------------------------------------------------------------------------

TRACKING:    S-74  --> already tracked; moving to its successors

SUCCESSORS:  Point ---> Interval ---> Point
SUCCESSORS:  S-74,. --> S-75,-    --> (S-41,.f)

ADV3
INSERTION:   S-75,- ==> (S-76,- S-77,. S-78,-)  [S-77,. will be reading state]
UPDATING:    S-77,.  (updating Q2 values from Nsim predictions)

PREDICTED:   State   Varname    Q2                Nsim
PREDICTED:   S-77,.  AMOUNT     [19.00 20.88]     [18.81 20.79]

-----------------------------------------------------------------------------
             State   Variable   Observed          Predicted         Similarity
             -----   ---------  --------------    --------------    ----------
SIMILARITY:  S-77    Time       [2.20 2.20]       [2.20 2.20]       1.000
SIMILARITY:  S-77    Amount     [19.19 20.37]     [19.00 20.79]     0.922
SIMILARITY:  S-77    Amount'    [-12.67 -11.93]   [-17.31 -7.60]    (range of Netflow)
SIMILARITY:  S-77    Amount''   [6.790 7.210]     inc               (qdir  of Netflow)
SIMILARITY:  S-77,.  ........................................... = 0.922
-----------------------------------------------------------------------------

TRACKED:     S-77,.
UPDATING:    S-77,.  (updating Q2 values from readings)
UPDATING:    S-77,.  (updating Nsim values from readings)
UPDATING:    S-77    Amount     [19.19 20.37] <-  [18.81 20.79] (updating nsim)

=============================================================================
CANDIDATES:  1 original, 1 retained, 0 rejected, 0 hypothesized, 0 new.
```

```
    1:  Drain = lo-blockage
        states = (S-77)
        similarity = 0.922   apriori probability = 0.07
        age = 1.20  [created at 1.10, highest similarity at 1.00]
==============================================================================
```

BIBLIOGRAPHY

[Abb90] Kathy Hamilton Abbott. *Robust Fault Diagnosis of Physical Systems in Operation.* PhD thesis, Rutgers University, May 1990.

[All84] James Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.

[BBd82] J. S. Brown, R. R. Burton, and J. deKleer. Pedagogical, natural language, and knowledge engineering techniques in sophie i, ii, and iii. In D. Sleeman and J. S. Brown, editors, *Intelligent Tutoring Systems*, pages 227–282. Academic Press, 1982.

[Ber91] Jared Daniel Berleant. *The Use of Partial Quantitative Information with Qualitative Reasoning.* PhD thesis, Department of Computer Sciences, The University of Texas at Austin, August 1991. Also published as tech report AI 91-163.

[BML89] Ivan Bratko, Igor Mozetič, and Nada Lavrač. *KARDIO: A Study in Deep and Qualitative Knowledge for Expert Systems.* MIT Press, 1989.

[Can86] James V. Candy. *Signal Processing: The Model-Based Approach.* McGraw-Hill Series in Electrical Engineering. McGraw-Hill, 1986.

[Can88] James V. Candy. *Signal Processing: The Modern Approach.* McGraw-Hill Series in Electrical Engineering. McGraw-Hill, 1988.

[CFP89] R. N. Clark, P. M. Frank, and R. J. Patton. *Fault Diagnosis in Dynamic Systems: Theory and Applications*, chapter 1: Introduction, pages 1–19. Prentice Hall, 1989. (editors: Ron Patton, Paul Frank, Robert Clark).

[Cla89] William J. Clancey. Viewing knowledge bases as qualitative models. *IEEE Expert*, 4(2):9–23, summer 1989.

[CS90a] J. T.-Y. Cheung and G. Stephanopoulos. Representation of process trends—part I. a formal representation framework. *Computers Chem. Engng.*, 14(4/5):495–510, 1990.

[CS90b] J. T.-Y. Cheung and G. Stephanopoulos. Representation of process trends—part II. the problem of scale and qualitative scaling. *Computers Chem. Engng.*, 14(4/5):511–539, 1990.

[Dav84] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24(3):347–410, December 1984. Also in *Qualitative Reasoning about Physical Systems*, pages 347–410, D. G. Bobrow (ed.), MIT Press, Cambridge, Massachusetts, 1985.

[Dav89] Randall Davis. Form and content in model based reasoning. In *Working Papers of the 1989 Workshop on Model Based Reasoning*, pages 11–27, August 1989. Detroit, Michigan.

[DeC90] Dennis DeCoste. Dynamic across-time measurement interpretation. In *AAAI-90*, pages 373–379. American Association for Artificial Intelligence, AAAI Press, July 1990.

[Den86] P. J. Denning. Towards a science of expert systems. *IEEE Expert*, 1(2):80–83, 1986.

[DF91] Richard J. Doyle and Usama M. Fayyad. Sensor selection techniques in device monitoring. In *Proceedings of the Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pages 154–163. IEEE Computer Society Press, April 1991.

[DH88] Randall Davis and Walter Hamscher. *Exploring Artificial Intelligence*, chapter 8: Model-based Reasoning: Troubleshooting, pages 297–346. Morgan Kaufmann Publishers, 1988.

[DK89] Daniel Dvorak and Benjamin Kuipers. Model-based monitoring of dynamic systems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1238–1243, August 1989. Detroit, Michigan.

[DK91] Daniel Dvorak and Benjamin Kuipers. Process monitoring and diagnosis: A model-based approach. *IEEE Expert*, 6(3):67–74, June 1991.

[dKW87] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, April 1987.

[dKW89] Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1324–1330, August 1989.

[DSA87] R. J. Doyle, S. M. Sellers, and D. J. Atkinson. Predictive monitoring based on causal simulation. In *Proceedings of the Second Annual Research Forum*, pages 44–59. NASA Ames Research Center, 1987.

[DSA89] Richard J. Doyle, Suzanne M. Sellers, and David J. Atkinson. A focused, context-sensitive approach to monitoring. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1231–1237. International Joint Conferences on Artificial Intelligence, Inc., August 1989.

[Dvo87] Daniel L. Dvorak. Expert systems for monitoring and control. Technical Report AI87-55, Department of Computer Sciences, The University of Texas at Austin, May 1987.

[FD90] David Franke and Daniel Dvorak. CC: Component-connection models for qualitative simulation – a user's guide. Technical Report AI 90-126, Department of Computer Sciences, The University of Texas at Austin, January 1990.

[FOK90] F. Eric Finch, O. O. Oyeleye, and Mark A. Kramer. A robust event-oriented methodology for diagnosis of dynamic process systems. *Computers & Chemical Engineering*, 14(12):1379–1396, 1990.

[For86] Kenneth D. Forbus. Interpreting measurements of physical systems. In *AAAI-86*, pages 113–117, Los Altos, August 1986. American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc.

[Fou91] Pierre Fouché. Abstracting irrelevant distinctions in qualitative simulation. In *Working Papers of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, pages 232–244, May 1991.

[FW89] Paul M. Frank and Jürgen Wünnenberg. *Fault Diagnosis in Dynamic Systems: Theory and Applications*, chapter 3: Robust fault diagnosis using unknown input observer schemes, pages 47–98. Prentice Hall, 1989. (editors: Ron Patton, Paul Frank, Robert Clark).

[Ger92] Janos Gertler. Structured residuals for fault isolation, disturbance decoupling and modelling error robustness. In *Proceedings IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, April 1992. Newark, DE.

[Ham91] Walter C. Hamscher. Modeling digital circuits for troubleshooting. *Artificial Intelligence*, 51:223–271, 1991.

[HHW84] James D. Hollan, Edwin L. Hutchins, and Louis Weitzman. Steamer: An interactive inspectable simulation-based training system. *AI Magazine*, V(2):15–27, Summer 1984.

[Ise89] Rolf Isermann. *Fault Diagnosis in Dynamic Systems: Theory and Applications*, chapter 7: Process Fault Diagnosis Based on Dynamic Models and Parameter Estimation Methods, pages 253–291. Prentice Hall, 1989. (editors: Ron Patton, Paul Frank, Robert Clark).

[Kay91] Herbert Kay. Monitoring and diagnosis of multi-tank flows using qualitative reasoning. Master's thesis, The University of Texas at Austin, May 1991.

[KB88] Benjamin Kuipers and Daniel Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 324–329. American Association for Artificial Intelligence, August 1988.

[Kel90] Richard M. Keller. In defense of compilation. In *Working Papers of the Second AAAI Workshop on Model Based Reasoning*, pages 22–31, July 1990. Boston, Massachusetts.

[Kit89] M. Kitamura. *Fault Diagnosis in Dynamic Systems: Theory and Applications*, chapter 9: Fault Detection in Nuclear Reactors with the Aid of Parametric Modelling Methods, pages 311–360. Prentice Hall, 1989. (editors: Ron Patton, Paul Frank, Robert Clark).

[Kot85] P. A. Koton. Empirical and model-based reasoning in expert systems. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 297–299, August 1985. Los Angeles, California.

[Kuh70] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* The University of Chicago Press, second edition, 1970.

[Kui86] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29(3):289–338, 1986.

[Lac91] Franz Lackinger. *Model-Based Troubleshooting: Qualitative Reasoning and Impacts of Time.* PhD thesis, Vienna University of Technology, December 1991. Christian Doppler Laboratory for Expert Systems.

[LCS+88] Thomas J. Laffey, Preston A. Cox, James L. Schmidt, Simon M. Kao, and Jackson Y. Read. Real-time knowledge-based systems. *AI Magazine*, 9(1):27–45, Spring 1988.

[Mal87] Donald B. Malkoff. A framework for real-time fault detection and diagnosis using temporal data. *Artificial Intelligence in Engineering*, 2(2):97–111, 1987.

[Mic82] James A. Michener. *Space.* Random House, Inc., 1982. This novel is a story about the American space program, fiction for the most part, but built around real events.

[MW89] Raymond C. Montgomery and Jeffrey P. Williams. *Fault Diagnosis in Dynamic Systems: Theory and Applications*, chapter 10: Analytic redundancy management for systems with appreciable structural dynamics, pages 361–386. Prentice Hall, 1989. (editors: Ron Patton, Paul Frank, Robert Clark).

[Ng90] Hwee Tou Ng. Model-based, multiple-fault diagnosis of time-varying, continuous physical devices. In *Proceedings of the Sixth IEEE Conference on AI Applications*, pages 9–15, 1990. Reprinted in *IEEE Expert*, 6(6):38–43, December 1991.

[Per84] Charles Perrow. *Normal Accidents*. Basic Books, Inc., New York, 1984.

[PFC89] Ron Patton, Paul Frank, and Robert Clark, editors. *Fault Diagnosis in Dynamic Systems: Theory and Applications*. International Series in Systems and Control Engineering. Prentice Hall, 1989.

[Rei87] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[Sca89] Ethan Scarl. Sensor failure and missing data: further inducements for reasoning with models. In *Working Papers of the 1989 Workshop on Model Based Reasoning*, pages 1–6, August 1989. Detroit, Michigan.

[SD87] Reid Simmons and Randall Davis. Generate, test and debug: Combining associational rules and causal models. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 1071–1078, August 1987. Milan, Italy.

[SPT86] Paul A. Sachs, Andy M. Paterson, and Michael H. M. Turner. Escort – an expert system for complex operations in real time. *Expert Systems*, 3(1):22–29, January 1986.

[TC86] Robert A. Touchton and Mike Casella. Reactor emergency action level monitor: a real time expert system. In *Instrument Society of America Convention*, October 1986. [REALM].

[Wel88a] Daniel S. Weld. Comparative analysis. *Artificial Intelligence*, 36(3):333–374, 1988.

[Wel88b] Daniel S. Weld. Exaggeration. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 291–295. American Association for Artificial Intelligence, August 1988. Saint Paul, Minnesota.

[Wid89] L. E. Widman. Expert system reasoning about dynamic systems by semi-quantitative simulation. *Computer Methods and Programs in Biomedicine*, 29(2):95–113, 1989.

[Wit87] A. Witken. Scale-space methods. In Stuart C. Shapiro, editor, *The Encyclopedia of Artificial Intelligence, Volume 2*, pages 973–980. John Wiley & Sons, 1987.

[WKAS78] S. M. Weiss, C. Kulikowski, S. Amarel, and A. Safir. A model-based method for computer-aided medical decision making. *Artificial Intelligence*, 11:145–172, 1978.

[WLN89] Lawrence E. Widman, Kenneth A. Loparo, and Norman R. Nielsen, editors. *Artificial Intelligence, Simulation & Modeling.* John Wiley & Sons, 1989.

VITA

Daniel Louis Dvorak was born in Terre Haute, Indiana on July 24, 1950, the son of John and Nona Dvorak. After graduating from Schulte High School, he entered Rose Polytechnic Institute (now named Rose-Hulman Institute of Technology) and completed a Bachelor of Science degree in electrical engineering in 1972. He then joined AT&T Bell Laboratories as a Member of Technical Staff and served in the Air National Guard. He attended Stanford University, graduating in 1974 with a Master of Science degree in "Electrical Engineering: Computer Engineering", and returned to work at Bell Laboratories.

His work over the years entailed a variety of design and programming projects in operating systems, computer networking and distributed computing. In 1984 he received the "Distinguished Technical Staff Award for Sustained Achievement" and began doctoral studies in September 1985 at The University of Texas at Austin.

Permanent address: 2S 455 Barclay Place
Glen Ellyn, IL 60137-6912

This dissertation was typeset[1] with LaTeX by the author.

[1]LaTeX document preparation system was developed by Leslie Lamport as a special version of Donald Knuth's TeX program for computer typesetting. TeX is a trademark of the American Mathematical Society. The LaTeX macro package for The University of Texas at Austin dissertation format was written by Khe-Sing The.