MODEL-BASED DIAGNOSIS OF COMPLEX, CONTINUOUS MECHANISMS

APPROVED BY
SUPERVISORY COMMITTEE:

This dissertation is dedicated to my family. I wish especially to acknowledge my mother, Dr. Sara Thompson Throop; her fortitude in completing her own dissertation when I was a youth set my steps on the path towards doctoral work. I wish also to recognize my father, William Rutherford Throop; his lifelong love of biology turned my young interest to the sciences.

MODEL-BASED DIAGNOSIS OF COMPLEX, CONTINUOUS MECHANISMS

by

DAVID RUTHERFORD THROOP, B.S. ChE.

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 IN COMPUTER SCIENCES

THE UNIVERSITY OF TEXAS AT AUSTIN
August, 1991

Acknowledgments

I wish first to acknowledge the help and guidance of my advisor, Dr. Benjamin Kuipers, who has labored mightily to help me transform myself into a scholar. Thanks to Dr. Janice Beyer, who gave me sound advice when the dissertation process seemed most perplexing. Thanks to the other members of my committee, Drs. Robert Simmons, Charles Chiu, Bruce Porter and David Himmelblau for their suggestions all along the way. Thanks to Dr. Sara Throop, Raman Rajagopalan, Adam Farquhar, Wood Wai Lee and Dr. Pearl Pu for their constructive criticisms of draft copies of this work. Again, thanks to Dr. Sara Throop for financial help throughout my graduate years. Thanks to Micki Rosenberger for moral support. Thanks to Lisa Jackman for encouragement. Thanks to NASA for their funding of the work of the AI Lab at UT. Thanks to Bess Sullivan, Marlene Bateman, Esther Anderson and Renee Loyless, the secretaries of the AI Lab.

DAVID RUTHERFORD THROOP

The University of Texas at Austin
August, 1991

Abstract

In diagnosis, when a hypothesis proposes a variable's value, several different lines of evidence may be considered; the different evidence must be arbitrated. The result of this arbitration consists of a single best estimate of the variable value and of a measure of that estimate's plausibility. The plausibility measure reflects the degree of agreement among the lines of evidence.

This report describes HEATX, a program for model-based diagnosis of nonlinear mechanisms with continuous variables. Previous work in model-based diagnosis has avoided arbitrating numeric evidence, often by representing continuous variables as discrete symbols (e.g., high, cold). Such restricted representation have had difficulty in diagnosing mechanisms with feedback or reconvergent fanout. HEATX represents numerical data explicitly in the hypotheses and in the inferencing procedures; it is thereby able to arbitrate evidence numerically.

HEATX uses both nonlinear numerical simulations and approximate linear models to perform diagnosis in the domain of heat-exchanger networks. The response of these networks to changes in their inputs is nonlinear; the networks also have feedback and reconvergent fanout. This dissertation introduces several novel techniques for diagnosing such networks. It interleaves the generation of complete fault hypotheses with several tests on partially formed hypotheses. Two of these tests are the *qualitative filter* and the *clustering filter*.

The *qualitative filter* analyzes the signs of gains between fault and symptom variables. The *clustering filter* constructs linear approximations to individual components and assembles these into a linear model of the network. It then uses the linear model to assess the consistency of a hypothesis. It does so by determining whether there is a value for the candidate fault variable which is consistent with the quantitative values of the symptom variables; the degree of agreement between the symptoms and best value for the fault variable is used to score the hypothesis. This filter is extended to multi-fault diagnosis, in which values for several fault variable may be estimated and judged simultaneously.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction to Diagnosis

> *A single heat-exchanger, working normally, cools a hot stream of liquid, using its heat to heat a second, cooler stream. The outlet temperatures of the streams are monitored; they are steady. A fault occurs: a furnace has malfunctioned. The hot stream's inlet temperature falls. There are two symptoms: both outlet temperatures fall. The fault is repaired and normal operation returns.*
>
> *A different day, another fault occurs. Inadvertently, a valve is partially closed; the hot stream's massflow drops. Again there are two symptoms: both outlet temperatures fall.*
>
> *Other days bring other faults. The exchanger receives too high a massflow in the cold stream; both outlet temperatures fall. Then something causes the cold stream's inlet temperature to drop. Once again, both outlet temperatures fall.*

Qualitatively, the symptoms are the same each time. The faults are not distinguished by the symptom's directions of change. But closer examination shows that the *ratio* in which the temperatures fall is different each time.

Each fault has a different profile. The fault profiles are complex. A fault's profile changes with the fault's magnitude; a small drop in the hot stream massflow will have a different profile than a large drop. But the profiles are distinct enough to guide diagnosis.

1.1 Motivating Diagnostic Reasoning

The overall goal of this research is to automate the diagnosis of complex mechanisms. This work has already been started by others, but the diagnosis of many important mechanisms remains beyond the state of the art. In particular, mechanisms containing feedback, nonlinear components and continuous variables present unresolved problems.

This dissertation seeks to automate diagnosis for such mechanisms. We proceed by building programs that use multiple representations of mechanisms to support different phases of the diagnosis: linear models to generate and select candidate hypotheses and numerical representations to test these hypotheses in detail. We pursue this approach in the domain of diagnosis for networks of heat-exchangers.

Several points make networks of heat-exchangers a good domain for research on diagnosis. First, such networks are interesting because they are representative of a large class of flow problems with continuous variables in which feedback and nonlinearity cause difficulties for automated diagnosis. Secondly, fault detection and estimation for networks of heat-exchangers by numerical computational methods is not straightforward, and diagnosing them is known to be difficult. Finally, the problem itself is of interest and is tractable.

This report presents advances in diagnosis, in variable estimation, and in the integration of simulation into diagnosis. Like other recent work in automated diagnosis, it uses models of mechanisms both to generate diagnostic hypotheses and to test them. It develops a much-overlooked aspect of such model-based diagnosis: it emphasizes that when a mechanism's variables are continuous, a complete hypothesis contains not only an *identification* of the fault variables, but also an *estimation* of the variable values. It further emphasizes that when a mechanism is nonlinear, this estimation is potentially difficult, and becomes a major part of the diagnosis.

The report describes HEATX, a model-based diagnostic program which implements these methods. The techniques of model-based diagnosis have previously been demonstrated on numerous mechanisms in which continuous variables were collapsed into discrete symbols. But there are important yet subtle diagnostic distinctions which are not represented adequately by discretized variables. HEATX extends model-based diagnosis to deal with problems using a fully numerical representation, while retaining the strengths of the model-based approach.

1.1.1 A Model-Based Approach to Diagnosis

Model-based reasoning and model-based diagnosis are methodologies for automation for representing, reasoning about and diagnosing mechanisms. In model-based diagnosis, a model of the correctly functioning mechanism is used to predict the mechanism's behavior. Discrepancies between these predictions and the observations of the mechanism are symptoms; the rest of the diagnostic task (hypothesis formation, refinement and testing) is driven by these symptoms.

Model-based reasoning developed in response to pressing problems in automated reasoning. Earlier methods encoded knowledge about how mechanisms failed, but did not encode knowledge of how correctly functioning mechanisms worked. Programs using these methods were unable to diagnose novel faults, or known faults which manifested in novel ways.

Knowledge which represents a mechanism's structure and the physical principles which determine that structure's behavior has been called *deep knowledge*. This contrasts with representations which link the names of faults to the names of symptoms, called *shallow knowledge*. Similarly, *structural* descriptions (based upon the physical structure of an object) can be represented independently of their particular use, and can therefore be used and reused more freely than *functional* descriptions (based upon the intended function of the object).

Deep knowledge, grounded in fundamental principals (conservation laws, current laws, mass balances) can provide explanations of novel faults. It can constrain the generation of spurious diagnoses, rejecting some hypotheses as impossible on physical grounds. And deep knowledge and structural descriptions offer the possibility of reusing knowledge: the data describing a mechanism's structure (particularly, its components and their connectivity), are available from design data and operating data.

1.1.2 The Extension of Model-Based Diagnosis to Continuous Variables

There are advantages to using model-based methods in diagnosis. However, difficulties are encountered when these methods are applied to complex mechanisms. The advantages, their difficulties and their solutions are discussed individually below.

By using a model, automated diagnosis gains the capability to *explain novel faults*. But in order to use numerical simulation models, the models must be provided with numeric inputs. The hypothesis generation techniques used heretofore in model-based diagnosis have not provided such numeric values. This dissertation integrates *variable estimation* into model-based diagnosis.

By using component-connection models, and by modeling those components structurally (rather than in terms of their function in a particular context), automated diagnosis becomes *grounded in fundamental principles.* However, in complex systems, these structural descriptions are constraints which are solved algorithmically, computing the component's behavior from its (possibly faulty) inputs. As diagnosis works in the opposite direction (from behaviors back to faults), it must invert the algorithms which satisfy these constraints, add these inverted algorithms to the component model, and assemble these inverted component models into inverted mechanism models.

The computation for the inverted algorithm may be harder and less exact than the initial algorithm: computing $f(x) = x^3$ is easy, computing $f^{-1}(x) = \sqrt[3]{x}$ is harder. No method that is both exact and completely general is known for such inversion. This dissertation presents a method using *linear approximations* to *invert component behavior*, and methods for correcting the errors introduced by the linearization.

Also by using component-connection models, automated diagnosis can *constrain hypothesis generation*, using the connectivity information of the model to trace causal pathways. Following these pathways, faults or sets of faults which could cause the observed symptoms are identifiable. But in mechanisms with feedback, non-directional components or reconvergent fanout, these pathways contain loops. This leads to situations in which most fault variables are causally connected to most symptoms. This leads in turn to a failure to constrain the hypothesis generation. This dissertation presents a *qualitative filter* and a *clustering filter* as methods which prune spurious hypotheses, and presents techniques which *interleave hypothesis generation and hypothesis testing* to further constrain the number of hypotheses generated.

By using existing models, automated diagnosis can *reuse knowledge*; one common class of models we wish to reuse are numerical simulations. This reuse can result in an economy of description: when knowledge of component behavior is reusable, there exists the potential for building up libraries of component descriptions, libraries whose descriptions can be used in several tasks. But, as we have already said, numerical simulation models compute behaviors from inputs; they do not compute inputs from behaviors. This makes them unsuitable for direct use in hypothesis generation. This dissertation presents methods for *moving between multiple models* in order to use the knowledge embedded in numerical simulation models. It also presents a way of generating *different models (some for hypothesis generation, some for testing) from a single specification*, retaining the advantages of model reuse: ease, consistency and task integration.

Methods of automated diagnosis must *arbitrate evidence*: chains of inference from several sources are arbitrated into a single combined conclusion. It is our belief that, in the difficult problems mentioned above, it is necessary to do this arbitration in the numerical domain. Consequently, the data structures which express the relation of one variable to another need to express the numerical relations between the variable values. Further, the diagnostic hypotheses (which are the arbitrated conclusions from the evidence) need to express both the value of that arbitration, and also the degree of agreement across the multiple lines of numerical evidence.

We hold that, if such expressiveness were included in the knowledge representation and in the propagation and inferencing routines, the resulting program could be powerful enough to reason correctly about the behavior of complex mechanisms involving feedback. We also hold that if such a program were written along the lines suggested by the advances in model-based reasoning, it would retain the virtues of model-based diagnosis: the ease in rapidly modifying the mechanism description and adding new information, the ability to diagnose novel faults and the ability to reuse models which had been developed for other purposes, such as design and simulation models.

1.1.3 Modeling Issues Addressed in This report

There are many aspects of modeling which research in model-based reasoning might address. It may be helpful to make an overview of the types of modeling question addressed in this report.

	Less Difficult		*More Difficult*
$\surd$	Single fault	$\surd$	Multiple fault†
$\surd$	Steady state		Dynamic behavior
$\surd$	Absence of noise		Presence of noise
$\surd$	Continuously present fault		Intermittent fault
	No feedback	$\surd$	Feedback present
	Invertible models of components	$\surd$	Non-invertible models
	Discrete variables	$\surd$	Continuous variables
	Qualitative information	$\surd$	Numeric information
$\surd$	Seek best explanations		Seek all explanations
	Expect complete data	$\surd$	Resiliant to missing data
	Behavior is function of inputs	$\surd$	Device has internal state
$\surd$	Deterministic models		Probabilistic models

$\surd$'s indicate problems and modeling issues addressed in this report.
†Hypothesis generation is limited to single fault; hypothesis filtering is extended

to multi-fault.

1.1.4 Claims of This Report

We propose advances in knowledge representation and automated reasoning about the behavior of mechanisms with continuous numerical variables. To reason accurately about such mechanisms requires

- an explicit representation of the numeric values for the variables in the knowledge representation used by the inferencing methods;
- that the inferencing methods reference these values;
- that the inferencing methods can propagate the quantitative effects of changes in these variable values both
  (a) from causes to effect, and
  (b) from effects to causes;
- that the inferencing methods can arbitrate among quantitative values when there are multiple lines of inference which determine a variable value;
- that the result of this arbitration provides both an estimate of the variable value and a measure of the plausibility of this estimate, based on the degree of agreement between the multiple lines of evidence.

Providing this expressiveness, both in the knowledge representation and in the inferencing methods, will support the correct diagnosis of complex, continuous mechanisms containing recycle and feedback, mechanisms which could not otherwise be diagnosed in model-based reasoning.

1.2 Terms Used in Diagnosis

The field of model-based reasoning has developed its own terminology, which we shall use. Since the meanings of some terms varies from use to use, we will present some of the central terms here.

A *diagnosis* is a plausible explanation for an unexpected behavior or feature of a mechanism. We stress that diagnosis explains *unexpected* behavior; behavior which is undesirable but expected (such as that of a fault-free mechanism subjected to a known overload) does not need to be diagnosed.

A *behavior* is a set of observable (or calculable) variables and variable values from a mechanism; it includes the values of the symptom variables and also of other observable variables which are within their nominal ranges. It may also include the values of variables which cannot be observed directly, but whose values can be derived from observable data.

A *nominal* value is an intended or expected variable value. The nominal value may be known from design data, or from sensors, or calculated from other variables which control it (or *should* control it.) A *symptom* is a variable whose value (either as observed or calculated from observations of variables it controls) has deviated by a significant amount from its nominal value.

A *mechanism* is something which has structure and whose behavior and state are the object of diagnostic attention. Different domains of diagnosis use different terms for this, including 'patient,' and 'system.' Heat-exchanger networks are not normally understood to be patients, and the term 'system' is often used to refer to the program for diagnosis itself. We have therefore exclusively used the term 'mechanism' in this regard.

Mechanisms are made up of *components*. Components have *variables*, and variables have values. In model-based diagnosis, we generally assume that the system is completely characterized by its components, their connectivity, and the component's variable values, (although Section 3.9 discusses exceptions to this view.) Some variables are independent; other variables values are determined by the values of the independent variables and the constraints of the model. In this report, we do not make a distinction between exogenous variables (variables which get their values from processes outside the mechanism) and the independent state variables of components. Instead, we have a special class of components, 'sources,' which provide the values of the exogenous variables in the model; this way, *all* variables are component variables.

The behavior of a mechanism is determined by (a set of) *constraints*. These constraints may be algebraic expressions enforcing relationships among component variables, or they may be encoded algorithms which calculate variable values. Constraints generally only relate the variables of a single component. The exceptions to this are constraints that describe the mechanism's connectivity, which relate the variables of two different components.

A *fault*, in our representation, is the departure of an independent variable from its

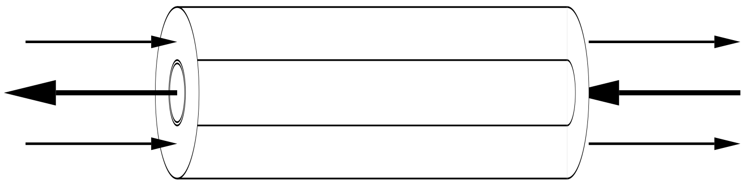


Figure 1.1: Counter-Current Heat Exchange

nominal value. In this representation, components are never 'broken.' The constraints that describe the component continue to hold, but the behavior that is predicted changes because the values passed to the constraints have changed.

The decision to represent faults this way is equivalent to having explicit fault models of components. It makes a commitment to knowing the ways in which components can fail, and encoding constraints in such a way that they apply to both faulted and unfaulted components. It may mean the introduction of variables whose nominal value is zero (such as a 'leak rate' for each pipe joint) and the inclusion of these variables in the constraints.

In a fault, an independent variable has departed from its nominal value. If the variable is continuous, this departure is a (numeric) change in the variable value, and the magnitude of this change is part of the complete description of the fault.

A *hypothesis* is an explanation (plausible or otherwise) of a mechanism's behavior. It consists of one or more proposed fault variables and values for these variables. The hypothesis claims that, if its fault variables took these values, the effect would be the observed behavior.

1.3 Introducing the Domain of Heat-exchanger Networks

We have pursued our approach to diagnosis in the domain of heat-exchanger networks. A heat-exchanger is a device for transferring heat from a hotter fluid stream to a colder one. Inside the heat-exchanger, the two streams are in thermal contact; the streams cannot intermix, but heat flows through a thin barrier.

1.3.1 Industrial Use of Heat Exchange Equipment

In industrial applications such as chemical plants and oil refineries, there may be dozens of fluid streams which need to be heated or cooled. A cool stream may be heated in several heat-exchangers between where it is drawn from storage and where it is delivered for processing (e.g., to a reaction vessel or a distillation column.) Similarly, the hot streams may exchange with several cool streams between coming from processing and arriving at storage. These hot and cool streams may tie together dozens or even hundreds of heat-exchangers into a network.

To understand the economic importance of such networks, consider their use in oil refineries. In order to get crude oil hot enough to distill, heat is supplied in three ways. The cold crude passes through a series of heat-exchangers, being warmed by the finished products (gasoline, asphalt, kerosene and so on) on their way to storage. It then passes to a series of furnaces (in which fuel is burned to heat it further) and distillation columns (where steam is used to separate the lighter oils.)

Heat-exchangers recover heat from hot product streams. Any heat that isn't recovered must be supplied to the cool stream by using more fuel in the furnace or steam in the column. More than half the heat supplied to the crude is recovered from the hot streams.

To refine a barrel of oil, a refinery burns between 0.5% and 1.5% of the barrel as fuel to supply the heat to distill the remaining oil. Without heat exchange, this would about double, increasing the fraction of the crude consumed in refining by at least 1%. Since virtually all crude is refined, this would amount to 1% of all the oil used in the world. Thus, the proper functioning of the heat-exchanger network is of great economic importance to a refiner.

1.3.2 Mathematical Modeling of Exchangers

Equations describing the behavior of heat-exchangers are available. A heat-exchanger's outlet temperatures are a function of the inlet variables (massflows and temperature); independent state variables (heat transfer coefficients) and other variables that describe the systems but can be assumed not to vary (for instance, the number of tubes in the exchanger, or data that describe the physical properties of the stream's density and specific heat.)

A heat-exchanger is not a linear device. The outlet temperatures cannot be accurately calculated as linear functions of these variables. The equations that govern the functional relation between the outlet temperatures and the input variables and independent state variables are the energy balance and the heat transfer equations. Introducing the terms

T is a Temperature,
c_p is a specific heat,
W is a mass flow rate,
U_o is the overall heat transfer coefficient,
Q is the rate of heat transfer,
A is the surface area,
$\overline{Lmtd}$ is the Log Mean Temperature Difference, defined by

$$\overline{Lmtd} = \frac{\Delta T_1 - \Delta T_2}{\log \frac{\Delta T_1}{\Delta T_2}},$$

where

$$\Delta T_1 = T_{HotIn} - T_{ColdOut}$$

$$\Delta T_2 = T_{HotOut} - T_{ColdIn},$$

then for a counter-current heat-exchanger, the energy balance is

$$\Delta T_{Hot} \cdot c_{p_{Hot}} \cdot W_{Hot} = -\Delta T_{Cold} \cdot c_{p_{Cold}} \cdot W_{Cold},$$

where

$$\Delta T_{Hot} = T_{HotOut} - T_{HotIn}$$

$$\Delta T_{Cold} = T_{ColdOut} - T_{ColdIn},$$

and the heat transfer equation is

$$Q = U_o \cdot A \cdot \overline{Lmtd}.$$

The energy balance uses single values for each specific heat c_p. In general, a fluid's specific heat is temperature dependent. For our present purposes, using a constant specific heat is accurate enough if we use the specific heats of the streams at their average temperature within the exchanger. However, we cannot know these average temperatures until we know the outlet temperatures.

The functional relationship between T and c_p is different for different fluids. Consequently, we have no general functional relationship (in closed form) giving the outlet temperatures T_{HotOut} and $T_{ColdOut}$ of a heat-exchanger. In practice, we calculate

T_{HotOut} and $T_{ColdOut}$ iteratively, estimating them initially, then calculating in turn $\overline{Lmtd}$, Q, the average c_p's, and a revised value for T_{HotOut} and $T_{ColdOut}$.

As we have said, the outlet temperatures are functions of the inlet and independent state variables. But instead of being able to express this relation functionally, we must calculate it iteratively. It is even more difficult to calculate the inputs from known values of the outputs. And this is just for a single component of a much larger network.

As we will discuss in Section 3.4.1, much of the work in automated diagnosis has assumed that the models of components (and the models of mechanisms composed from them) are *invertible*; that is, given the constraints governing the component and the outputs from the component, the inputs can be calculated straightforwardly.

This assumption does not hold for heat-exchangers and for many other important devices. This property reflects the complexity of heat-exchanger behavior. It forces us, if we are to reason about systems of such devices, to encode reasoning methods more powerful than have yet been captured in automated diagnostic systems.

1.4 Organization

This dissertation is organized in five chapters. This chapter has briefly introduced the methods of model-based reasoning and model-based diagnosis, and the domain of heat-exchanger networks. Chapter 2 presents background material, including an overview of model-based reasoning and of other current work in automated diagnosis. Chapter 3 presents our contributions to the theory of model-based diagnosis; it presents a mathematical development of the qualitative filter and the clustering filter, and an extension of the clustering filter to multiple fault diagnosis. Chapter 4 presents the details and results from an implementation of this theory; describing the diagnostic program HEATX and its performance on several sets of problems. Chapter 5 discusses the implications of this work and presents ideas for future research. Examples of the methods developed here are distributed through the individual chapters.

Portions of this work appeared in an earlier form in [Throop, 1987].

Chapter 2

Diagnosis in Perspective

2.1 Background

We suggested in Section 1.1 that the current state of the art in automated diagnosis is still unable to reason robustly about some important mechanisms, and we mentioned the difficulties reasoning about mechanisms which contained feedback cycles. We will begin this chapter by reviewing why researchers had turned away from both associative representations and from conventional mathematical representations for diagnosis and were looking to ground automated diagnosis in models.

Although some mechanisms can be diagnosed adequately with the methods of model-based reasoning already published and well understood, other mechanisms exhibit behaviors that cannot be treated by these methods. Such mechanisms may include, in combination, such features as:

- variables in the mechanism which are continuous, rather than discrete;
- the presence of feedback loops among the components;
- the presence of components whose response is not linear; and therefore
- the presence of components whose behavior is not invertible.

We have chosen to explore diagnosis in the domain of networks of heat-exchangers because such networks exhibit these difficult features.

2.2 Associative Systems in Diagnosis

Early work on diagnosis considered diagnosis as a single, more-or-less uniform activity. The General Problem Solver [Newell and Simon, 1963] [Newell, 1981] treated diagnosis as a state-space search problem, to be solved by using search theory and weak methods, [Newell, 1973]. As problem spaces and problem specifications grew larger, expert systems began using domain knowledge to manage heuristic search in diagnosis. This knowledge, encoded as empirical associations between causes, intermediate states and symptoms, allowed computer programs to deliver expert-level diagnosis of difficult problems. Such programs, (such as MYCIN, [Shortliffe, 1976] and PROSPECTOR [Gaschnig, 1979]) showed impressive power, but they proved to be difficult to modify and gave poor results on problems containing novel faults or features that had not been specifically encoded into the knowledge base. The EMYCIN [Davis *et al.*, 1984] program further separated the domain knowledge from the inference capability; this made it easier to extend the knowledge base but underscored the difficulties inherent in encoding all diagnostic knowledge as associations between symbols. In particular, quantitative data and knowledge about continuous variables did not fit into a simple associational scheme. The VM program [Fagan *et al.*, 1984] extended the coverage of MYCIN style rules by mapping quantitative measurements into qualitative categories: (very low | low | acceptable | high | very high.)

Work in automated diagnosis during the 1970's used numerous approaches to knowledge representation, (see Brachman and Levesque [Brachman and Lesveque, 1985] for a collection reviewing such approaches.) Many of these works shared an underlying representation that was (or was equivalent to) as set of rules of the form

```
IF Predicate1(Object1)  THEN Predicate2(Object2)
```

Such representation came to be called *associative representation* [Brachman and Lesveque, 1985, p 95].

Although the associative representation proved powerful at recognizing combinations of faults whose manifestations had been individually entered into the knowledge base, it dealt less well with any faults displaying novel symptoms.

By the end of the 1970's, associative representations had proven their worth. Rule-based expert systems were beginning to succeed in commercial applications. Rule-based and frame-based expert systems provided an easy way to encode and understand descriptions of real-world mechanisms.

Associative systems separate knowledge from program code and from the data that describes a particular problem, [Newell, 1981]. This separation provides much of the power and usefulness of such systems. Still, even as these systems were beginning to see widespread use, theoreticians began to notice shortfalls in the ability of the associative representation to represent important knowledge.

Hayes [Hayes, 1979] argued that such a representation is fundamentally impoverished: it could not gracefully encode much of the knowledge that people use to do trivial reasoning about physical objects. He called for an effort to encode the knowledge of the physical world that *all* humans (not just experts) have and use everyday. He titled this call the *Naive Physics Manifesto.*

2.2.1 Drawbacks of Associative Models for Diagnosis

Many researchers heeded Hayes' call. As they built novel knowledge encodings they began to notice other shortcomings of the associative representations, both from a theoretical view and a practical one. Randall Davis [Davis, 1985] articulated several such drawbacks.

Associative systems encode knowledge about faults that have been observed, or at least been explicitly anticipated. Experts have (hopefully) seen all of the faults that a particular mechanism manifests; their experience can be captured in the form of If/Then rules. But there is a drawback; if there are possible faults which have *never* been observed, one would still hope that a diagnostic system could recognize them if they ever did occur.

Davis pointed out that traditional rule-based expert systems failed when presented with novel modes of failure, or novel symptoms. Related to this, there seemed to be no systematic way of generating a list of all of the plausible failure modes for a mechanism; knowledge engineers just had to trust the experts to remember all the failures. This drawback became particularly acute when trying to build a diagnostic system for a newly built mechanism; no history exists for such a mechanism, so there are no experts and therefore no experiential knowledge to encode in rules.

Working in the domain of electronic circuits, Genesereth [Genesereth, 1985] noted that the knowledge encoded in the rules was specific to the mechanism being diagnosed. Rule-based expert systems separated the knowledge from the specific data about a particular case. But the separation was not clean enough to carry over to a similar mechanism. The rulebase that had been so laboriously built up for one electronic circuit could not be reused on the next one.

Not only was the knowledge overdependent upon the data, it also was too tightly tied to the inferencing code. A rulebase about a mechanism is, in some sense, a model of the mechanism. Other detailed models of mechanisms exist in comput-

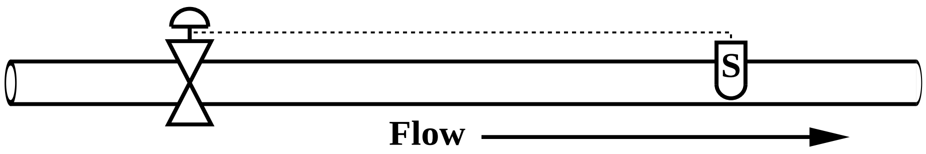


Figure 2.1: A Feedback Mechanism Not Well-Encoded by Rules

ers: design information, data bases, simulation models. Chandrasekaran and Mittal [Chandrasekaran and Mittal, 1982] noted that it would be desirable to combine such representations, so that the knowledge necessary for diagnosis would be derivable in some natural way from (and consistent with) the other extant models; Davis and Genesereth made similar suggestions. But the rule-based representation alone did not naturally support this.

Rulebases were seen as difficult to construct, [Waterman, 1986]. No clear methodology emerged that allowed a knowledge engineer to go directly and systematically from the known structure of a mechanism to an encoding in a rulebase.

The difficulty in encoding the behavior of a mechanism is a result of what rule-based expert systems encode, or fail to encode. The rulebase captures a *shallow* description of the mechanism. It describes the observable behaviors of the mechanism, but it does not capture the mathematical relationships within the mechanism that generate those behaviors. Further, it only poorly captures the relationships among the components that make up the mechanism, and it is through these components that causal events flow, connecting faults to the observable behavior. DeKleer and Brown [de Kleer and Brown, 1984] and Klein and Finin [Klein and Finin, 1987] note that this descriptive failure leads to diagnostic systems that are "unable to deal with circumstances even slightly different than those explicitly anticipated."

Associative systems do not encode feedback systems well. Consider the simple feedback mechanism shown in Figure 2.1. The individual components might be encoded in a few rules. For the valve

```
Pressure(Low) => Flow(Low)      Pressure(High) => Flow(High)
Signal(Low) => Valve(Wide)      Signal(High) => Valve(Narrow)
Valve(Narrow) => Flow(Low)      Valve(Wide) => Flow(High)
```

For the sensor and its transducer

```
Flow(High) => Signal(High)      Flow(Low) => Signal(Low)
```

Although these are reasonable encodings of the individual components, it is unclear what the system will decide when trying to diagnose abnormally high flow: high flow seems to be caused by a wide valve, which is caused by a low signal, which is caused by low flow, which could be caused by low pressure.

What these rules do *not* capture is the subtlety of the relationship: in response to high pressure, the flow increases, but not as much in comparison to the flow that would occur without the feedback control of the valve.

Traditional rule-based systems do not encode ordered values. A continuous variable can be made discrete by associating symbols with divisions of its range; e.g., a variable may be allowed to assume the values *very-low* | *low* | *normal* | *high* | *very-high*. But this is not enough. Although the standard forms of associative systems provide a formalism for arbitrating belief about conflicting evidence, these formalisms lack relevant knowledge about how to arbitrate ordered evidence. If one line of evidence indicates that the variable is *very-low* and another line indicates that it is *normal*, the two lines of evidence are not arbitrated to a belief that the variable is *low*. Rather, the system holds evidence for two contradictory beliefs (for values of *very-low* and *normal*) and no evidence for *low* at all.

2.2.1.1 Qualitative Algebras of Discrete Variables

This failing can be addressed by supplying qualitative algebras to the system. Systems based on such discretizations have shown improved performance, when compared to traditional rule-based representations [Fagan *et al.*, 1984] [Scarl *et al.*, 1985] [Scarl *et al.*, 1988] [Malin *et al.*, 1990] [Glass *et al.*, 1990]. But discrete qualitative algebras have their own problems:

- Such algebras are inherently ambiguous. What is the sum of *low* and *very-high*?
- The algebras quickly get very complex, because different variables will be divided into ranges with different units. For instance, if the heat flow of a stream is the product of the temperature and the mass flow, we must define the product of *cold* and *high*.
- Certain divisions in the qualitative algebras are inherently capricious. We can divide pipe sizes into *tiny* | *small* | *normal* | *large* | *huge,* but our choice of these ranges will depend strongly on the particular mechanism we are studying.

The difficulties cause by discretization of continuous variables have been addressed by the use of *quantity spaces*, (proposed by Hayes, [Hayes, 1979] [Hayes, 1985], and systematized in Forbus [Forbus, 1981] [Forbus, 1984] as QPT, and Kuipers, [Kuipers, 1984] as QSIM.) This work will be discussed further in 2.4.6.

2.2.1.2 Numerical Information in Associative Systems

Associative representations do not encode numerical information well; there is no obvious place to locate numerical information within the formalism. These representational shortcomings often force us to cast problems in bizarre ways. In the manual for a popular expert system shell, an example problem involves reasoning about the cost of a used car. The example makes a simplifying assumption: used cars either cost $1000, or $2000, or $3000. This is not a good representation of such costs.

An additional difficulty in the associative representation of numerical information occurs when evidence conflicts. When multiple lines of evidence are joined, they must be arbitrated together, [Cohen, 1985]; the reasoning system must detect if the evidence agrees or conflicts and must synthesize the multiple lines of evidence into a single judgment. EMYCIN supplied *certainty factors* as a general procedure for resolving conflicting evidence about discrete variables; later work advanced other such methods. No such general method emerged for the arbitration of conflicting numeric data; if one line of evidence indicated that a temperature was 15.1° high and a second line indicated that it was 15.3° high, no automatic procedure existed to determine whether this was good agreement or profound contradiction. Users could, of course encode their own arbitration schemes, but these never became general parts of the representational framework.

This is not to say that numerical computation cannot be done by associative systems. Such systems have always provided 'backdoors,' ways of escaping from the rule language into the underlying language (usually LISP) to do calculations, [Melle,]. But calculations *by* the system are not the same as calculations *in* the system. Such an escape is outside the theory of associative systems and returns us to the same problems with arbitrary code that originally led to the emergence of rule-based systems.

2.3 Introducing Model-Based Reasoning

We have seen how associative reasoning systems have difficulty expressing important knowledge about their domains. Because it had proven difficult to encode complex mechanisms in associative representations, researchers began using more robust models to build automated diagnostic systems. Realizing that associative representations usually captured only empirical associations between the inputs and the outputs of components of mechanism, Davis [Davis, 1985] proposed reasoning directly from components' structure. This approach, of *reasoning from first principles*, replaces the associative descriptions of components with constraint sets. This work also showed a clean separation of hypothesis generation from hypothesis testing. Davis showed that such an approach could produce correct diagnoses of "bugs that display novel symptoms," (p. 390 in [Davis, 1985].) He also observed that a model-based approach is more easily modified as the domain changes.

In the early 1980's, work in automated diagnosis focused on ways of achieving better competence in the knowledge representation. Many AI researchers built diagnostic systems which used explicit models of the mechanisms they analyzed. In 1986, Davis gave an overview of this work in an invited talk at AAAI, (later revised with Hamscher and issued as [Davis and Hamscher, 1988].)

This body of work was referred to in different ways; much of the early work was called "diagnosis from first principals," [Davis, 1985] [Genesereth, 1985] [Reiter, 1987]. In time, the modeling issues in such work came to the fore, and the use of models was seen to be important in other automated reasoning tasks beside diagnosis. The field adopted the terms "model-based reasoning" and "model-based diagnosis" for the approach; these are the terms we adopt in this present work.

2.3.1 The Meaning of "Model-Based Reasoning"

What has the term "model-based diagnosis" come to mean? Obviously, model-based diagnosis uses a model to do diagnosis. But the term *model* means many things, and in some sense *any* diagnostic system encodes a model. The term denotes a more specific approach:

- Model-based diagnosis uses a model of the correctly-behaving mechanism. It is *not* just a collection of fault models.
- Model-based diagnosis is based on the comparison of observation and prediction. The discrepancies between the observations and the predictions are the symptoms, and these drive the diagnostic task.
- Model-based diagnosis is symptom directed. This in contrast to "diagnostics:" procedures by which the inputs to the mechanism are systematically manipulated in order to drive the system through a large number of states. The focus

in model-based diagnosis is on determining explanations for known symptoms, not on discovering new symptoms.

- Model-based diagnosis takes inferencing about a condition as far as it can with the information that it has. This is in contrast with diagnostic methods which focus optimizing the next question asked or the next probe placed. As Davis points out, "Inference is 'free,' while measurement is often quite expensive. ...It's well worth it to 'think as much as you can' before taking another measurement," [Davis, 1985].

- Model-based diagnosis is generate-and-test. Different aspects of the model may be used in the generation and testing phase. But generation can produce some hypotheses which are not consistent with the model and the symptoms; the model is then run to falsify these hypotheses.

- Models in model-based diagnosis are component models, [de Kleer and Brown, 1984]. The knowledge about the mechanism is structured in a way patterned after the mechanism's own structure.

- In model-based diagnosis, the knowledge engineering task is the descriptions of the behavior of components, rather than the behavior of entire mechanisms. Component behavior is encoded in such a way that the overall behavior of the mechanism is emergent from the component behaviors, their connectivity and the inputs.

This approach puts diagnosis on the same theoretical footing as the modeling theory it uses; for good or ill, depending on the models we use. Fortunately, large parts of modeling theory are on very firm ground, and the process of how humans use cognitive models to reason has already received a great deal of attentions. (For a sense of this work, see Gentner and Stevens, [Gentner and Stevens, 1983]. For an overview, see Norman in the same volume, [Norman, 1983].)

To quote Davis and Hamscher again, "A fundamental presumption behind model-based reasoning for diagnosis is the notion that if the model is correct, all the discrepancies between observation and prediction arise from (and can be traced back to) defects in the device," [Davis and Hamscher, 1988]. Ng makes a finer point, that each fault in the mechanism corresponds to, and is localizable to, some fault in an individual component, [Ng, 1990].

2.3.2 The Promise of Model-Based Diagnosis

What does adopting a model-based approach gain us? Section 2.2 gave an accounting of the shortfalls that caused researchers to move away from the purely associative approach to diagnosis. These, then, are the advantages of the model-based approach:

- A model-based approach can diagnose novel faults and novel symptoms. The possible failures of individual components are known. Component behaviors can be combined together (in the model) in ways which generate novel mechanism behaviors (in the model), and which can match novel observed behaviors (from the physical mechanism.)
- Model-based diagnosis provides a systematic way of generating faulty mechanism behaviors. Components are simple enough that encoding their behavior is relatively easy. Once the behaviors and failures of components are correctly described, the complex manifestations of a fault in the larger mechanism are generated by the modeling process.
- The models in model-based diagnosis encode deep knowledge of the mechanism. The descriptions of the individual component behaviors are much easier to ground in the "deep" knowledge of the domain: fundamental laws of physics, conservations laws and so on.
- Model-based diagnosis allows a natural way to reuse models that have been built for other purposes, such as design and simulation models. Current practice in design and simulation also uses component models, and there is usually a reasonable way to map from the design models into the diagnostic models.

The last item highlights a tacit definition often assumed in descriptions of model-based diagnosis:

- A model-based diagnosis program uses a *single* model for diagnosis: a single model that (potentially) it shares with other programs, written for other tasks.

We speak to the implications of this tacit definition in Chapter 3.

2.3.3 Generate and Test

Model-based diagnosis uses generate-and-test [Newell, 1973] to explain unexpected behavior in mechanisms. It generates hypothesis by some suitable means, and uses a model (or models) to test them, [Davis and Hamscher, 1988]. Both the generation and the testing phases may be subdivided, and the phases can be interleaved.

Variables may also be discrete, i.e., have a domain of a relatively small number of symbolic values, one of which is the nominal value. For discrete variables, the identification of the variable plus a hypothesized, non-nominal value constitute the complete hypothesis. If the discrete values of the variable are ordered, a less complete hypothesis may be formed: an identified variable plus a hypothesized direction of change ($+$ or $-$) from the nominal value. When variables are boolean, just the identification of a variable (suspected of being non-nominal), constitutes a complete hypothesis.

For continuous variables, a complete hypothesis requires the identification of the variable plus a numeric variable value or an estimate of the variable value. Like ordered discrete variables, a less complete hypothesis (identification plus directions) can be formed.

For some *continuous mechanisms*, (that is, mechanisms with continuous variables), the fault variable value can be quickly and exactly calculated from the observations. In diagnosing such mechanisms, calculating a variable value is a minor detail in hypothesis formation. In other diagnoses, calculating an estimate of the variable is a complex task in itself; it is the task of *variable estimation*.

The work on model-based diagnostic systems has not generally focused on the task of variable estimation. The variables of most mechanisms studied have been boolean or discrete. The research that has considered continuous mechanisms has, for the most part, either studied mechanisms in which the variable values could be calculated directly, or it has discretized the continuous variables, (although see the discussions of the works of D'Ambrosio, [D'Ambrosio, 1990], and Patil, [Patil, 1981], [Patil *et al.*, 1984], below.)

2.3.3.1 Fault Models in Hypothesis Generation

A fault model is a model of how something behaves when it is faulty. One of the difficulties with traditional rule-based expert systems is their reliance on fault models of mechanisms; any listing of the possible faulty behaviors of a complex mechanism is usually incomplete. A system which relies upon such a listing will be unable to account for novel faulty behaviors which do not appear in the list. For this reason, model-based diagnosis has generally not used fault models of entire mechanisms. However, many model-based diagnostic systems use fault models of components. A system which uses component fault models assumes (a) these models cover all possible faulty behaviors the component can exhibit, and (b) all system faults can be localized to one or more component faults. The system thereby limits its diagnoses to mechanism behaviors which originate in known component faults. This is not as restrictive as it first sounds. Many novel behaviors are unusual manifestations of well-understood component faults; Section 2.4.7.1 will examine such a case.

Because individual components are simpler than assemblies of components, it is more feasible to account for all of the possible faults of a component (or a class of components) than for an entire mechanism. Models of components may incorporate constraints reflecting physical principals (energy conservation, Kirchoff's law) which cannot be violated even by faulty mechanisms. Such principals constrain the possible faulty behaviors and it is prudent to use such knowledge when it is available. If a system does not constrain the possible behaviors of faulty components, it may generate bizarre explanations of behaviors. Struss and Dressler [Struss and Dressler, 1989] make the point clearly in their example of a lightbulb circuit. In it, a battery lights three lightbulbs, connected in parallel; the observed symptoms are that two of the bulbs are not lit. One hypothesis is that the battery is dead, and the lit bulb is broken in such a way that it lights when no voltage is present.

However, there are some advantages to reasoning without fault models of components. It is sometimes possible to demonstrate that, no matter what the behavior of an individual component, it could not have caused the observed symptoms. Such a demonstration exonorates the component; this can be much more efficient than working through a long list of fault models for a component and showing that each one is inconsistent with the symptoms. There are also classes of fault in which constraints derived from fundamental physical principals are violated in unexpected ways. One such class is mistakes in assembly; in the lightbulb circuit, if a careless electrician had installed a (lighted) candle where the specifications called for a lightbulb, the fault hypothesis would be true. Another class of fault is one in which a component takes on some unexpected property. Ungar [personal communication] suggested the following example: catalyst may come loose from a reactor and lodge in a downstream heat-exchanger, causing the heat-exchanger to act as a reactor. If this happens, the component and energy constraints around the reactor (encoded in a form which did not include any terms reflecting chemical reactions) will no longer be true.

2.3.4 Generation – Variable Identification

Automated diagnosis has used many methods for generating candidate fault variables. This section will discuss the criteria by which we judge the suitability of a generation method for a particular diagnostic task, and then discuss several popular methods of hypothesis generation.

Davis and Hamscher [Davis and Hamscher, 1988] remind us that it is desirable that methods for hypothesis generation have several properties.

> Classical AI wisdom tells us that a good generator should be complete, (i.e., capable of producing all the plausible hypotheses), non-redundant (i.e., capable of generating each hypothesis only once), and informed (i.e., able to produce few hypotheses that ultimately prove to be incorrect.)

Because most work in model-based diagnosis has not considered variable estimation, it has largely treated hypothesis generation as a matter of identification.

2.3.4.1 Methods without Models

We have said that model-based diagnosis uses a model to do diagnosis. However, a model-based system may not use a model for all phases of the diagnosis; non-modeling approaches can be used in generation, and still be tested with a model.

One way to generate fault candidates is to recognize that symptoms are similar to symptoms seen in previous diagnoses. This approach need not have a model of the mechanism, just some previous history of successful diagnoses, or the encoded knowledge of a diagnostician who has dealt with such problems. The strength of such an approach is in its ability to reuse prior experience; many problems arise repeatedly and give stereotypical manifestations. We consider three methods that can supply testable hypotheses without using models: fault dictionaries, associative systems and neural nets.

Coverage is a difficult issue for such dictionaries if the number of possible faults is not small; the number of simulation runs is, in the general case, exponential in the number of faults. Himmelblau, [Himmelblau, 1978, p. 280], discusses methods of placing limits on the number of simulations needed for good coverage. Fault dictionaries are not generally tolerant of missing data; fault classification performance degrades badly when no value is available for supposedly observable variables. They also degrade quickly when presented with incorrect data. A similar criticism can be made for the related technique of fault trees.

Section 2.2.1 discussed traditional rule-based systems and the diagnostic and representational difficulties they engender. They can, however, be very good at recognizing previously seen faults. For this reason, large implemented systems (which will be described in Section 2.4) have generally augmented their hypothesis generation with heuristic rulebases.

A related approach is to compile the detailed information in a model into an associative rulebase; Venkatasubramanian and Rich, [Venkatasubramanian and Rich, 1988], discuss an implementation of such an approach in which the domain is chemical processing equipment.

Neural nets have also been used to identify faults in chemical processing equipment. Hoskins and Himmelblau, [Hoskins and Himmelblau, 1988] trained a neural net to recognize six different classes of fault in a series of continuous stirred-tank reactors. Venkatasubramanian *et al* [Venkatasubramanian *et al.*, 1990] trained a neural net on single-fault failures and showed that it was often capable of diagnosing multiple-fault failures when the multiple faults were composed of single faults from the training set. Leonard and Kramer [Leonard and Kramer, 1990] improved the backpropagating algorithm used to train the net; although the claim that with such an improvement, "novel patterns can be recognized," they appear to be speaking of novel combinations of known faults, with novel combinations of known symptoms. They do not appear to be speaking of novel single faults, or novel manifestations.

Each of these methods, lacking any explicit model of the mechanism which they diagnose, will have difficulties in diagnosing novel faults. This same difficulty limits their use as hypothesis generators; the testing model can't test that which is not generated. If the mechanism itself is novel, or if it is reconfigured frequently, then the experience needed to build a heuristic or neural net diagnostic program will not be readily available. Nonetheless, such programs can be an invaluable augmentation to any other generation method.

2.3.4.2 Constraint Suspension & Dependency Backchaining

Dependency backchaining and constraint suspension are two complimentary methods for generating and filtering fault candidates. Dependency backchaining provides a way of determining what components are causally upstream of all symptoms. It does this in conjunction with a simulator that records the way in which values are derived, as well as the values themselves. This is often accomplished using a *truth maintenance systems* or TMS. (Section 2.3.4.4 will discuss set-covering methods which essentially preserve constraint suspension but do not use the same propagation methods.) Components are modeled as sets of constraints on values, and faults are modeled as retractions of these constraints. This approach does not use fault models; no new constraints are added that describe how the component behaves when broken.

The dependency backtracking suggests which components are causally connected to the faults. The constraint suspension provides a quick filter that makes the generation more informed. Simmons and Davis [Simmons and Davis, 1987] describe a system in which the dependency records, in addition to directing the hypothesis generation, guide the debugging of hypotheses that are rejected when tested. Davis and Hamscher [Davis and Hamscher, 1988] provide a good description of dependency backchaining and constraint suspension, and offer this caution,

> The power of hypothesis testing and its ability to infer symptoms are dependent on the power of the propagation machinery. Current constraint systems are "local" in the sense that they propagate values through one component at a time, at each step solving one equation in one unknown. This style of propagation can stall when it encounters situations requiring more sophisticated algebra (e.g., solving two equations in two unknowns.) Such situations are relatively common in domains with nondirectional components and can arise in domains with directional components in structures that have *reconvergent fanout* (i.e., a signal that branches and then rejoins at the inputs to a component.)

Hamscher, [Hamscher, 1988] commenting on these methods also warns "[The problems of diagnosing without explicit fault models of components] become evident when diagnosing a system with feedback or high connectivity between its components. If only knowledge about correct behavior is used then almost any discrepancy can be accounted for by the failure of any component."

Dependency-backchaining methods rely upon the ability to invert values through components. We will discuss in Section 3.4 how this reliance fails in some mechanisms.

2.3.4.3 Incremental Construction

Multiple fault hypotheses can be generated one fault at a time using *incremental construction*. The symptoms are explained as well as is possible by the best-fitting single-fault hypothesis. The unexplained symptoms, called *residuals*, are treated as a new diagnostic problem. The approach works best when the symptoms of individual faults do not overlap. It encounters progressively more difficulty when symptoms do overlap, when symptoms interact synergically, when symptoms cancel each other, and when conditions interact to evoke symptoms that neither evokes singly.

In cases in which faults do not strongly interact, incremental construction can provide rapid diagnoses of multiple faults. Pople wrote a program, INTERNIST-I [Pople Jr., 1982] to address such problems of symptom interaction by using a *partitioning heuristic* to judge which symptoms should be addressed by the first hypothesis and which could be set aside for later explanation.

Patil [Patil, 1988] discusses an example that typifies how the partitioning heuristic fails when symptoms *do* strongly interact. He describes ABEL, a program which addresses this failing by the construction of a hierarchical *patient specific model.* This is a causal model in which, at the lowest level, conditions are grounded in quantitative numerical models of biochemical and physiological processes. These models are invertible:

> Causal links ... specify the relationship between the severity, duration and other relevant aspects of the cause and the effect nodes, ... Furthermore, reasoning may be carried out in the forward or the reverse direction; a cause may be used to predict the effects or an effect used to deduce the necessary severity and duration of a cause.

2.3.4.4 Set Covering

Set covering is another approach to variable identification for multiple fault diagnosis. It is the most thorough of the methods we will discuss, but it purchases that thoroughness at the cost of some deep computational problems. In this usage, a set cover is a set of faults which could, jointly, explain the symptoms in a behavior.

Set covering was proposed by Reggia *et al* **[no cite yet]**. It was formalized by Reiter [Reiter, 1987], who introduced the *minimal set cover.* Greiner *et al*, [Griener *et al.*, 1989] published a correction of this formalization, and deKleer *et al* [de Kleer *et* *al.*, 1990] introduced the *kernel diagnosis* to correct some of the theoretical difficulties in the way that the minimal set cover had been defined.

Set covering uses Reiter's DIAGNOSE algorithm to find the minimal cover, using *conflict sets*. A conflict set is a set of components such that it is inconsistent to believe that all components in the set are normal. DIAGNOSE starts with a conflict set containing all the components in the mechanism, removing components which do not help explain the symptoms, until only *minimal* conflict sets remain. Minimality, in this usage indicates, that if any component were removed from the conflict set, then it would no longer *be* a conflict set. That is, it would be consistent to believe that all the remaining components in the conflict set were normal.

To execute, DIAGNOSE must repeatedly check the consistency of a set of components. In the general case, consistency checking is NP-complete. However, in some specialized domains consistency checking can be performed in reasonable amounts of time. Ng [Ng, 1990] extended DIAGNOSE to consider qualitative models of continuous systems, and used QSIM [Kuipers, 1986] as a consistency checker.

The discussion of constraint suspension, above, noted problems which that technique had when considering systems with feedback, non-directional components and reconvergent fanout: when any single suspected component is causally upstream from all components, then any fault hypothesis seems to explain all symptoms. Hamscher attributed this to reasoning without specific fault models.

This same problem occurs with a different manifestation in the set covering approach. When many single components are causally upstream from all the symptoms, then all the minimal covers become unary: multiple fault hypotheses don't get generated when single faults appear to explain everything. We will return to a consideration of these difficulties in Section 3.12.

2.3.4.5 Non-AI Variable identification

We have noted that the AI work in automated diagnosis has generally focused on variable identification and ignored estimation. Outside AI, the trend has been the opposite; much of the work assumes that the list of suspect fault variables is small, and uses variable estimation (usually called 'parameter estimation' there) to make the final fault determination from a short list of suspects.

Model-based diagnosis imposes some specific requirements for methods which determine a mechanism's variable values. The mechanism must be represented in such a way that a computational model of it may be assembled from a description of its components and their connectivity. The diagnostic methods must conform to the knowledge representation. The work in variable estimation outside AI has not been greatly concerned with these issues. However, some of the representations used for variable estimation in control and systems theory appear to be adaptable to a component connection approach. For an introduction to the issue in this area, see [Sorenson, 1980]; for an extension to nonlinear problems see [Ross, 1990].

The study of *data reconciliation and rectification* has addressed the issue of diagnosis in a problem-space in which there are a large number of possible faults. Mah [Mah, 1990, 385ff] considered the approaches to rectifying mass, energy and component balances, both those caused by measurement error and those caused by gross error. He reviewed three different tests for detecting gross errors and discusses methods of fault variable identification which can be used in conjunction with these methods. ([Mah, 1990] is also interesting for its coverage of other modeling and reasoning issues. Although there are many other books addressing somewhat the same material, Mah represented many of the problems in terms of their data structures and examines the information-theoretic issues in the problems. His choice of representations should make his coverage of these issues of special interest to researchers in model-based reasoning.)

2.3.5 Generation – Variable Estimation

Chapter 3 will discuss a procedural problem in model-based diagnosis: given the description of a nonlinear mechanism, how can it be best modeled under the restrictions of composability (the model should be built up from component descriptions) and invertibility (the model should allow the identification of fault candidates when symptoms are given). That chapter will present a method for using linear component models for fault identification and for initial estimation.

Modeling nonlinear mechanisms with linear models introduces error. Even when fitting a linear model to a fault-free mechanism, the best-fitting variable estimations may report that many of the independent parameters are not at their nominal values; the error in the model appears as errors in the parameter estimates. Further, when modeling a mechanism with a single fault, the best-fitting linear model may vary several input parameters to produce a behavior with the observed symptoms.

The attribution of the effect of a single fault thus gets smeared over several candidate faults. However, parsimony leads us to prefer single-fault hypotheses to multi-fault ones, even when the multi-fault hypotheses fit the linearized model somewhat better.

It is also the case that two-fault hypotheses are preferable to three-fault, and so on. Reasoning with a linear approximation of a nonlinear mechanism is simpler when identifying and estimating a single variable (*point estimation*) than several variables (*joint estimation*).

Other methods available for automated diagnosis include several methods of variable estimation. Himmelblau [Himmelblau, 1978, Chapter 5] explored variable estimation for several different models, including models in which the constraints are algebraic equations, ordinary and partial differential equations and transfer functions.

2.3.6 Testing

Section 2.3.3 noted that a hypothesis about a continuous mechanism could be partially complete, specifying the sign of the fault parameter's deviation but not its magnitude. Such a hypothesis cannot be passed to a hypothesis tester that expects continuous variables to have numeric values. However, it may be possible to do some limited testing on the incomplete hypothesis; there is a computational saving for any hypothesis which is rejected before it is completed. Testing is thereby interleaved with generation; Hamscher [Hamscher, 1988] provided examples of such interleaving in model-based diagnosis.

Results of testing can also inform further hypothesis generation. Simmons and Davis [Simmons and Davis, 1987] described a method in which hypotheses were tested using consistency checking which builds up a dependency tree. If a contradiction was detected, the tree was used to debug, generating an improved hypothesis.

Testing can also refine an estimate; outside of AI, 'hypothesis testing' has generally referred to statistical methods which involve manipulation of the mechanism to produce additional data, [Himmelblau, 1978, p. 45] [Ross, 1990, p. 7]. Although the approach in this dissertation involves testing with additional computation instead of testing by additional data-gathering, the statistical techniques for hypothesis testing are of great potential relevance.

Hypothesis testing involves the comparison of predictions and observations. Broadly, when diagnosing a physical object, there are two methods of testing a hypothesis. One involves doing computation to generate additional predictions about the object and its behavior, and matching those predictions against existing observations. The second method involves taking additional observations and matching them against the existing predictions. When computation is cheap and measurement is not, the approach of 'think as much as you can' is preferred; see the discussion at Section 2.3.1. This is the approach taken in this dissertation. We note that there is existing work in model-based diagnosis on methods for determining the 'best next observation;' deKleer and Williams [de Kleer and Williams, 1987] describe the use of minimum entropy methods.

2.4 Aspects & Approaches to Model-based Diagnosis

We have described the general insights that support model-based diagnosis. Applying these insights when approaching an actual diagnostic problem inevitably raises a host of questions which shed further light on the general nature of the diagnostic task. Some of these approaches, reviewed in the following sections, have enlightened the approach that we offer in Chapters 3 and 4. As we will address the domain of heat-exchanger networks in those chapters, we will pay some extra attention to work that addresses topics in chemical engineering and chemical processes.

2.4.1 Large Implementations

Several large model-based diagnostic systems have been built. As this review shows, NASA has been the most important driving force behind pushing model-based reasoning to the application stage.

The CONFIG project (at NASA Johnson) develops a modeling tool for building discrete event simulations for reasoning about different configurations of engineered systems. By discretizing the numeric values into qualitative descriptors, it transforms the description of the behavior of such systems from continuous behaviors into behaviors that can be studied using discrete event simulation. This allows the possible predictions of malfunctions, [Malin *et al.*, 1987] [Malin *et al.*, 1990].

TEXSYS, (developed at NASA Ames), embodies a hybrid approach to qualitative and temporal reasoning to advise the thermal control system aboard the space station, [Glass *et al.*, 1990]. It addresses several difficult issues, including a large scaleup to a system with hundreds of complex, nonlinear components, noisy data and real-time performance.

2.4.2 Monitoring and Control

Several of these implementations of model-based reasoning involve reasoning about temporal behaviors for monitoring and control.

Faultfinder is a fault-monitoring and diagnosis expert system, (developed at NASA Johnson.) Its purpose is to assist a flight crew in fault monitoring, fault diagnosis and recovery planning, [Abbot *et al.*, 1987] [Schutte *et al.*, 1987] [Abbot, 1988]. The work integrates reasoning in an associative model with a deeper causal model, with the emphasis being on fast response to failure.

PREMON is a predictive monitoring project at JPL, [Doyle *et al.*, 1991]. The project addresses the task of focusing upon the most important details of a model, insomuch as PREMON's domain is space platforms with thousands of sensors. A development group led by Doyle has developed SELMON, a causal-model based system that does sensor selection.

LES, (NASA Kennedy), is an expert system for the handling of liquid oxygen, [Scarl *et al.*, 1985] [Scarl *et al.*, 1988] [Jamieson *et al.*, 1984]. It is a real-time systems using both rule-based and frame-based models of the domain. It particularly addresses the issues of sensor failure.

The systems in these monitoring projects each use combinations of causal models and calls to numerical routines for determining (or estimating) variable values. Other work has focused on monitoring using qualitative models. Interesting systems built in this manner include the work of Dvorak [Dvorak, 1987], and Dvorak and Kuipers [Dvorak and Kuipers, 1989], which built a qualitative simulation to reason about alarms, using the QSIM program to model the process. Similarly, Kay [Kay, 1991] described a method for dynamically recomputing the qualitative bounds around a qualitative behavior, given bounds on the qualitative constraints and on the initial conditions; they consider applications which include alarm limits which vary as process conditions change.

Boeing Computer Services Artificial Intelligence Center is developing an environmental control and life support system for NASA Marshall. The program incorporating both model-based reasoning and associative reasoning, [Center, 1990].

2.4.3 Approximate Reasoning and Comparative Analysis

Often, when predicting the possible behaviors of a mechanism, a purely qualitative description is insufficient to limit the possible behaviors, and an exact numerical value for all variables is unavailable. Raiman [Raiman, 1986] pointed out that, oftentimes, simply knowing the numeric values within an order of magnitude was enough to disambiguate predictions, and presented a methodology for making such inferences. Mavrovouniotis and Stephanopoulos [Mavrovouniotis and Stephanopoulos, 1989] extended these methods, implementing the result in an ATMS, [de Kleer, 1986].

Capturing the comparative behavior of mechanisms with feedback is still somewhat of an open problem in automated diagnosis. Weld [Weld, 1988] has investigated the problem for mechanisms described adequately by qualitative (non-numeric) descriptions. But work remains to be done.

2.4.4 Reasoning About Feedback

Section 2.2.1 discussed difficulties in representing a feedback system in a traditional rule-based system. In addition to not having a way of representing numerical information in the rules, such systems also suffer from a style of inference that is entirely local; recall Davis and Hamscher's remarks in Section 2.3.4.2 about how local propagation fails when trying to solve more than one equation in one unknown. As those remarks indicated, this is a failing that some model-based systems also share.

Reasoning about feedback necessitates reasoning about more than one variable at a time. For this reason, systems that deal well with feedback have comparatively been latecomers to model-based reasoning. Widman and Loparo [Widman and Loparo, 1988] note

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

Feedback is ubiquitous in natural systems. But its presence can make diagnosis extremely difficult.

There is a large body of mathematical analysis of feedback; it is the greater part of the field of process control, (Weber [Weber, 1973] is a good introductory text; see DeKleer for a discussion of the term "feedback," [de Kleer, 1985].) Variable estimation for mechanisms involving feedback is a broad area; for a sense of the scope of some of its application in geology, hydrology, population dynamics, epidemiology and other fields, see [IFA, 1978].

The techniques of these fields are gradually being incorporated into model-based reasoning. But the practice of model-based reasoning has not made a wholesale incorporation of these techniques. This has not happened both for technological and methodological reasons.

The emphasis in model-based reasoning has been on building models from brief descriptions of mechanisms, with models assembled automatically from listings of a mechanism's components and a specification of their connectivity. The technologies for describing feedback systems involve large amounts of algebraic manipulation, (finding the Laplace transform of a mechanism from its components' transfer functions is an example.) Model-based reasoning systems have been slow to use automatic algebra programs.

This has been partly due to problems with these programs, and is changing as these programs improve. But it is also due to the fact that the model construction in the feedback technologies still involves a degree of human judgment. For instance, the introduction of pure delay into a Laplace-transform model greatly complicates the algebra. Pipes introduce a pure delay; this delay is typically brief and is omitted from the model. The approximations are not difficult to encode; the judgment of when to use them is.

Practitioners of model-based reasoning have also been reluctant to use some available approximations, such as making linearity assumptions. The ability to automatically reason generally about loosely specified mechanisms is one of model-based reasoning's principal claims. Linearity assumptions run contrary to that spirit. Indeed, committing to a linear model could prevent a diagnostic system from explaining novel behaviors which arise from the interactions of nonlinear components.

The model-based approach to feedback has involved reasoning about it explicitly, rather than encoding any of the feedback technologies into a black-box and using it as an oracle. Model-based reasoning systems that have dealt with feedback include MIDAS [Kramer, 1990], Ng's work in qualitative diagnosis [Ng, 1990], and deKleer's program for reasoning about circuits, EQUAL, [de Kleer, 1985]. These have either represented the feedback as an explicit object with its own attributes (rather than, leaving the feedback implicit in the connectivity graph) or abstracted it into a representation in which cyclic interactions do not occur, (such as abstracting several series electrical resistors into a single resistance.)

2.4.5 Representation Languages and Libraries for Chemical Engineering Domains

Several groups have built or are building large modeling systems which seek to integrate the advances in AI representations with established methods for modeling large systems of chemical processing equipment. Although not all of these groups' primary aim is diagnosis *per se*, they address some of the same issues of including numerical information in an AI representational framework.

A group led by Lyle Ungar in the Chemical Engineering Dept of the University of Pennsylvania has undertaken modeling a large number of classes of chemical processing equipment and chemical and physical phenomenon, [Grantham *et al.*, 1990], [Grantham and Ungar, 1990]. The work focuses on troubleshooting and uses Forbus's QPT as a representational language for its models (see Section 2.4.6.2). The group is building up large "component libraries" of the kinds of components and processes found in chemical processing plants. They have reported advances in collapsing large sets of generated behaviors so that only relevant aspects of behavior are shown; their system automatically filters out undesired detail of the model. In [Grantham and Ungar, 1991], they discussed work in comparative analysis, using the same system. They make an interesting point in [Vinson *et al.*, 1991]:

> In process engineering, however, device models are not as straight-forward as in electrical engineering. Chemical devices (units) tend to be much more complex than electrical devices, partly because they handle multi-component inputs rather than just electrons. Also, chemical process units are produced as needed and customized, causing 'black box' models to vary for every application, whereas electrical components are mass-produced and so have uniform characteristics.

The group is developing DISARM, a modeling systems using Forbus's QPE. In hypothesis generation, they use a heuristic ordering of discrepancies to be explained, based on deciding which candidate faults are "farthest upstream" with cutpoints to break feedback loops; discrepancies cause the model to change.

Kramer and his students Finch and Oyeleye at MIT are developing a causal modeling program, MIDAS, [Kramer, 1990] [Oyeleye, 1990] [Finch, 1989]. A major focus of the work has been the development of modeling formalisms capable of representing a wide variety of plant behavioral knowledge. MIDAS uses both quantitative and qualitative representations, and particularly addresses difficulties encountered when reasoning about feedback loops.

Also at MIT, Stephanopoulos is developing MODEL.LA, a modeling language for process problems. [Stephanopoulos, 1990][1] Of particular note in this work is the representation of dynamic behaviors as series of bounding triangles; the method preserves certain critical details (such as sharp spikes and sudden transitions) even when data are aggregated over time periods orders of magnitude larger than the time scale of the details.

A group at CMU is developing ASCEND, an object oriented modeling language to assist in the design process, [Piela *et al.*, 1991]. The object-oriented approach particularly provides support for hierarchical representations of problems, including representing designs themselves in hierarchies of designs. The system represents component knowledge declaratively, rather than procedurally, supporting its use across a variety of tasks.

2.4.6 Qualitative Representation & Simulation

It may be difficult, when practicing model-based diagnosis, to determine at what level of detail to model. If we reason about a mechanism with specific numerical values, the behaviors we generate may not be generalizable to scenarios which use only slightly different variable values; we may reject correct hypotheses because of an unfortunate choice of values that we take to be typical. It is especially difficult to capture the behavior of mechanisms with variables that change continuously over time in associative representations. [McDermott, 1982] proposed a temporal logic to better capture events and processes.

Several researchers have built systems that can describe continuously changing variables in qualitative representations. Two of these efforts produced programs that have been widely used by other researchers. [Kuipers, 1984] produced a language for *qualitative simulation* called QSIM. [Forbus, 1984] advanced his *qualitative process theory*, or QPT.

The work presented in this report does not use either QSIM or QPT. However, consideration of each of these representations has greatly advanced our our under-

[1] there will be two additional citations to this work

standing of the place of numerical information in diagnostic reasoning. In particular, long experience with QSIM formed the ideas in the qualitative filter (which will be discussed in Section 3.6.1.)

We have already mentioned the use of qualitative simulation in the discussion of several implementations of model-based diagnosis.

2.4.6.1 Diagnosis using QSIM

Section 2.2.1.1 mentioned Kuipers' QSIM as a formal approach to the difficulties encountered in using qualitative algebras. We have also mentioned the use of qualitative simulation in the discussion of the work of Ng, Dvorak and Kuipers, and Weld; each of these diagnostic programs used QSIM to build the simulation model.

QSIM models are constraint-based. Typically, a single constraint will correspond to a single component. Bareiss and Farquhar [Bareiss and Farquhar, 1986] built fault models of components by substituting constraints describing faulty components for the constraints describing the normal behavior. Ng [Ng, 1990] did diagnosis without fault models, by withdrawing the constraint corresponding to the fault candidate and seeing if any behavior corresponding to the observations could be produced.

Roddis, at MIT, in her dissertation developed CRACK, a system which used QSIM to predict the growth of cracks in metal bridges, [Roddis, 1988]. CRACK was not component based, but offered a specific qualitative description of a fault development. The program moved between qualitative and quantitative data representations by having a single underlying data structure that both representations shared, so that important events in the qualitative simulation could be directly translated into quantitative values.

QSIM can correctly simulate a variety of feedback mechanisms in which multiple constraints must be simultaneously satisfied. The discussions above on feedback, Section 2.4.4, and on constraint suspension, 2.3.4.2, noted the tendency of local propagation methods to fail to solve problems with multiple constraints with multiple unknown variables. QSIM combines Waltz filtering, a local propagation method, with *global constraint filtering* to solve such problems.

Its success in this is attributable to several things. The Waltz filtering in QSIM is more sophisticated than the propagation methods used in some other systems. QSIM represents quantity spaces in such a way that a variable value can be partially restricted when complete information has not yet been generated. For instance, a value can be represented as ranging over several landmark values and the intervals between them, and yet still restricted from ranging below zero or up to infinity. When propagation stalls, QSIM combines constraints pairwise. Finally, if there are multiple solutions to a set of constraints, QSIM generates each of the solutions where other systems would stall.

The approach is still not sufficient for all of the inferencing in the system. QSIM builds qualitative representations of the higher order derivatives of functions in some of its constraints. It uses these higher order derivatives to eliminate certain spurious predictions of behavior. Waltz filtering and pairwise constraint satisfaction are not powerful enough to use this information; QSIM represents these higher order derivatives as algebraic expressions and uses a small automatic algebra system to solve them, [Dalle Molle, 1989a] [Kuipers and Chiu, 1987] [Chiu, 1988] [Kuipers *et al.*, 1991]. Dalle Molle [Dalle Molle, 1989b] used QSIM to model several dynamic chemical process systems, including linear and nonlinear processes, multivariable systems and a process under PI control.

- Reference Fouche. [Fouche and Kuipers, 1990]

2.4.6.2 QPT – Qualitative Process Theory

We have already mentioned Forbus' QPT as a representational system for qualitative modeling. QPT has been implemented as a modeling system QPE (qualitative process engine). The work with QPT has focused particularly on issues of model-formation: how to recognize the existence of entities (flows, connections, causal paths) and how to automatically encode the causal relationships as qualitative constraints. The qualitative representations in QPT are useful for describing continuous variables and continuously-changing processes. Several of the papers from this group have dealt with domains which overlap the one we choose in this dissertation. [Skorstad and Forbus, 1989], *Qualitative and Quantitative Reasoning About Thermodynamics*, describes the SCHISM program and its use. [Collins, 1989] describes a qualitative approach to modeling problems in thermodynamics. [DeCoste, 1990] applies the QPE to monitoring problems; the work presents an interesting examination of the asymptotic complexity of certain types of hypothesis formation.

Forbus's work on how to compose a model from a view of the processes active in them is useful in thinking about how to compose components into mechanisms. In this dissertation, we take the model of the process as a given; if the work were to be applied to a mechanism which was frequently reconfigured (by, for instance, the opening and closing of valves) the QPT approach could be a useful method of generating models for diagnosis.

2.4.7 Model-based Diagnosis and Other Methods

It may be enlightening to compare model-based diagnosis to three other similar techniques: discriminent analysis, cluster analysis and fault dictionaries.

2.4.7.0.1 Discriminent analysis – When we know a complete set of faults for a mechanism, and have a large amount of information on the way they manifest, we can use discriminent analysis. There can be overlap in these manifestations, (that is, in the typical symptoms of these faults, and in other traits correlated to the faults.)

The task here is not to classify a novel set of symptoms. Rather, it is to classify a "typical" set of symptoms in a field in which the manifestations of different faults are very similar, [Anderberg, 1973].

2.4.7.0.2 Cluster Analysis – When we do not know what the faults are, cluster analysis can help us sort out patterns in a collection of cases. What is given is the large sample of individual cases, each one with its own set of symptoms (a set of traits to be explained or classified), and indicators (traits which need not be explained, but which may further the explanation.) It is often unclear at the outset which indicators are of any help in the process and which are irrelevant. The task is to divide the cases up into useful categories, to find similarity metrics by which the population can be divided up into distinct populations with meaningful differences, [Anderberg, 1973].

2.4.7.0.3 Fault Dictionaries – When we have a good model of the system, we can run it through a large number of scenarios, and catalog the results. (We do not necessarily need a computer model; physical models are often used, too.) Then, when our mechanism fails, we can match the behavior it displays against one we generated on our model, [Park, 1979]. The task is to exploit our knowledge of the model so that we do not have to run the power set of all of our fault variables in order to get a useful classification, and to sort our results so they fit into some reasonably compact accounting.

It is expected that we will be diagnosing the same mechanism many times, and that it will not be reconfigured in such a way that the fault dictionary needs to be recreated. Typically, in generating such a dictionary, it is assumed that each fault variable is adequately covered by being represented as having just a few possible values, (such as *low* | *medium* | *high*); the possibility that a distinctly different behavior would be evoked midway between *low* and *medium*, and that such a behavior would be unknown to the modeler, is not addressed.

2.4.7.1 The Comparison to Model-based Diagnosis

Model-based diagnosis is good for diagnosing novel faults. When are novel faults seen? When a mechanism is diagnosed for the first time, everything seen is novel. When the inputs to the mechanism are at extreme values, faults may manifest in novel ways. A model-based approach can diagnose these faults because it builds up the novel behavior mechanisms from the known behaviors of components.

Consider the presentation of a novel manifestation of a fault, as described in [Kuipers and Kassirer, 1984]. Nephrotic syndrome is a medical condition in which a patient's kidneys malfunction. The kidney normally maintains a balance between the Starling equilibrium (which determines the balance between water in the plasma and water in the tissues), and a second equilibria (which controls the total amount of sodium and water in the body). In nephrotic syndrome, both of these equilibria are shifted to new stable points, changing the quantity of sodium and water. The usual result of this is edema; that is, the patient retains up to 50 pounds of excess water, with great swelling of the legs and abdomen.

Now consider the case of one patient who appeared to have something amiss with her kidneys: there were several symptoms which were consistent with nephrotic syndrome, but no edema, which is normally a hallmark of the syndrome. An interview revealed the patient to be a strict vegetarian who consumed virtually no sodium in her diet. In this case the physician made a diagnosis of nephrotic syndrome, because his knowledge of the equilibria allowed him to model the consequences of 'nephrotic syndrome' plus 'no sodium in diet.' Under these inputs, the model predicted the observed symptoms and did not predict edema; thus justifying the diagnosis.

How would this be handled in the each of the three methodologies just presented?

Discriminent analysis wouldn't diagnose it. Nephrotic syndrome is strongly associated with the presence of edema. The case would have failed to match either nephrotic syndrome or the syndromes with which it is easily confused. There is no way that missing feature (edema) can be "explained away" in the analysis.

Cluster analysis would have been inappropriate; the diagnostic task is not to recognize a new syndrome, just to classify a novel presentation into a known disorder.

If this case *had* been in a set used to generate a classification scheme for kidney disorders, it would have been an odd outlier.

Fault dictionaries would have been unlikely to help. It would have been unlikely that this particular set of inputs would have been picked to generate an input (and remember, a test of "low sodium" wouldn't have generated the behavior; it would have had to have been "drastically low sodium.") Davis and Hamscher [Davis and Hamscher, 1988] observe that decision trees, a related technology, suffer a similar shortcoming.

Model-based reasoning is able to explain novel behaviors at the mechanism level because it can generate them from the well-understood behaviors of individual components. Of course, diagnosticians in many fields are capable of explaining novel behaviors. What model-based reasoning offers is a way of representing their knowledge in a computer.

2.5 The Task of Model-Based Diagnosis

Diagnosis is a complex task. It is practiced in many, many domains. Many different fields have their own literature on the diagnostic task. Model-based diagnosis is, in its way, just one approach among many others.

The diagnostic methods in other fields are grounded in decades or even centuries of practice. Many of the idiosyncrasies of those methods take advantage of aspects of reasoning not easily applicable to other fields.

Model-based reasoning is not grounded in each of these fields. No one has attempted the task of surveying all of the different fields which practice diagnosis and then synthesizing the findings into a general protocol. (Or, if anyone is attempting it, no word has been heard back from them, and none is expected soon.)

Yet, first for the associative approaches that were advanced in the seventies and now for the model-based ones, the AI methods in diagnosis are offered as something *general*. They are offered as advances in the diagnostic process which are applicable across the many different domains in which diagnosis is done.

What basis is there for this claim? What is special about the job being attempted?

Diagnosis in almost all domains has been profoundly affected by the advent of computers. A pressing task in each of these fields is that of routinizing the cognitive process of diagnosis so that it can be encoded into a computer. It is always a hard task. Model-based reasoning has something general to offer here, not because it has made a synthesis from all fields of diagnosis, but because it addresses a struggle that cuts across domain lines. That struggle is the task is the encoding of diagnostic knowledge for use on computers.

Model-based diagnosis speaks to recurring problems in such encoding schemes:

- The difficulty of systematizing the knowledge.
- The difficulty in verifying it.
- The inability of such systems to diagnose problems only slightly different than the ones that were anticipated,
- The frustration attendant upon trying to use the knowledge already encoded in the computer for other reasons.

Each domain has its literature on advanced ways to solve particularly difficult problems. Model-based diagnosis has not yet incorporated many of these esoteric methods, and may therefore look like a naive approach to the experts in those domains. But the emphasis here has not been on *encoding* such knowledge. The emphasis is on finding ways that such knowledge *may be encoded*, and encoded quickly, transparently, generally and reusably.

Chapter 3

Model-Based Diagnosis for Complex Mechanisms

3.1 Our Principles for Diagnosis

Conceptually, there are three levels of things that a complete fault hypothesis says about a continuous variable –

- Variable identity. The hypothesis asserts that a particular variable has changed from its nominal value, "The feed temperature is abnormal."
- Qualitative estimate. It asserts a direction that the variable has changed, "The feed temperature is high."
- Numeric estimate. It asserts a numeric estimate of the variable's change, "The feed temperature is 17° high."

This last point has not been comprehensively addressed in model-based reasoning. Representing numeric information in hypotheses, and reasoning with it correctly, has motivated the development of our view of diagnosis. We briefly describe the central principles of this view here, developing them fully in the rest of the chapter.

Hypotheses about continuous variables consist of two parts: variable identification and variable estimation.

In forming a hypothesis, we first select a system variable that, we suspect, has departed from its nominal value. This is variable identification. With continuous variables, the hypothesis must contain a second part – an estimate of that the magnitude of the deviation. This is variable estimation.[1]

[1]The techniques we discuss will estimate the magnitude of the deviation in the fault parameter value, not the fault parameter value directly. For brevity, when it will cause no confusion, we will sometimes refer to this as "estimating the fault."

When fault and symptom variables are continuous, and when symptoms can be linked back to a candidate fault by a series of quantitative constraints, then each symptom makes a prediction – an estimate of the fault.

Even in a simplified representation, we can estimate the gain between the candidate fault variable and the deviation in an observed variable (a symptom.) We can use that symptom and that gain to estimate the fault. That estimate will be (approximately) correct *if* the candidate fault is, indeed, the cause of the symptom.

Multiple symptoms provide qualitative support for a hypothesis when their estimates agree in sign.

Each symptom makes a prediction of the deviation in the fault variable; that prediction is a signed number. Sign agreement across multiple predictions supports the hypothesis.

Conversely, sign disagreement contradicts a hypothesis. A candidate fault variable may be causally linked to two symptoms, but must decrease from 'nominal' to cause one symptom and increase to cause the other. Then the hypothesis that a deviation in this variable has caused both symptoms is implausible.

The agreement between symptoms in their quantitative estimate of a fault parameter's value is a useful measure of the plausibility of the hypothesis.

When there are multiple symptoms, there will be multiple estimates of the fault variable. The center of these estimates is the joint estimate of the fault variable's actual value. If these estimates cluster tightly together around the joint estimate, the fault hypothesis is more plausible than if the estimates diverge widely. When comparing multiple hypothesis, we can select those hypotheses with tighter clustering as more plausible, and pursue them first.

The invertibility of a knowledge representation plays a crucial role in its use in diagnosis.

Less exact representations (including linear approximations, qualitative representations and associative representations), are most useful in hypothesis formation and selection, where their very simplicity facilitates reasoning from effects (symptoms) to causes (faults).

More exact representations (including nonlinear numerical models) are most useful as a further check on the plausibility of a fault hypothesis.

When a plausible hypothesis has been formed, its fault variable has been identified and estimated. This can provide an input to run a non-invertible model, such as a nonlinear numerical simulation. When this model is run, it predicts values for the observable variables. If the agreement of this prediction with the actual symptoms is perfect (or within the error of the model) then the model supports the hypothesis in both its identification of the fault variable and its estimate. When the agreement is imperfect, this more exact model may support a revision of the fault variable estimate. Alternatively, it may provide evidence that the variable identification is incorrect – that no value of the fault variable could explain the symptoms.

3.2 Continuous Variables

As Section 1.2 explained, each possible fault is associated with some variable in the simulation model. If that variable is continuous, a complete fault hypothesis must include both an identification of the fault variable and a value for its magnitude. In simple mechanisms the magnitude may be calculated directly from the observation data. For more complex mechanisms, the value will be an estimate.

Continuous mechanisms (that is, mechanisms with continuous variables), are ubiquitous. When continuous mechanisms are diagnosed by traditional associative systems, the hypotheses formed will be incomplete; they will include no value for the fault variable or, at best, a discretized, qualitative value.

This is a shortcoming of traditional rule-based and frame-based expert systems, which use associative representations. It is also a shortcoming of any model-based diagnostic system in which the only underlying model is an associative one.

If associative models do not supply numerical estimates, what shall we use for hypothesis generation? At first glance, it might seem that we could generate them directly from numerical models, such as simulation models. Sections 3.4 through

3.5 will examine the pitfalls of such an approach. After these sections, we offer our own approach – building approximate numerical models and using them as *generating model* to generate complete hypotheses, then testing these hypotheses with more exact *testing models*.

3.3 Reasoning with Single Models

One of the advantages mentioned in Section 1.1.2 for model-based reasoning is an economy of description. Model-based systems, it is claimed, can reuse models.

Mechanisms are modeled for many other purposes; there are advantages in using the knowledge that is encoded in these existing models, rather than building a diagnostic model from scratch.

The advantages include

- Ease. We save ourselves work by not duplicating effort already done.
- Consistency. Using one model for two tasks assures that both tasks are reasoning about the same object. Changes made to the model in the course of doing one task are accessible to the other; we need not worry that the change failed to propagate.
- Task integration. If the diagnostic system is available to the design system, a designer may inquire about what failures are possible. He may propose undesirable behaviors of the mechanism and then allow the diagnostic unit to discover faults which could cause such behavior.

There is however, a big problem.

Most reused models cannot adequately support diagnosis. When they cannot, the best we can hope to do is to use them for *part* of the diagnosis, and augment them with other models.

Most models built for other purposes are not suitable for hypothesis generation because they are not *invertible*. Given the inputs, they compute outputs, but given the outputs, they cannot compute inputs. Their non-invertibility is not an artifact of the modeling language. Rather, it is a consequence of the complexity of the mechanism itself. This non-invertibility makes their reuse in hypothesis generation infeasible.

This infeasibility motivates a change from the definition of model-based diagnosis given in sections 2.3 and 2.3.2. In this present report, we do model-based diagnosis using multiple models. We use one model for hypothesis generation, another for testing.

3.4 Multiple Models and Multiple Representations

Both hypothesis generation and hypothesis testing use models. Why use two separate models? Why not just use a good model for generating the hypothesis, and save the trouble of testing? The answer is that the most exact models are not useful for hypothesis generation.

Many models have an inherent directionality; a nonlinear model will usually be non-invertible. To generate hypotheses, we must reason back through the model from observed symptoms to candidate faults. This backwards tracing violates the directionality of such models. If we insist on using only one model for diagnosis, that model will either be unable to capture important aspects of the behavior of many nonlinear mechanisms, or it will not support reasoning from symptoms back to faults. Instead, since model-based diagnosis is inherently generate-and-test, we will use different models for the generation and testing phases.

The multiple models we use are distinctly different – these are not just two distinct models written in the same modeling language. Rather, these models have different degrees of expressive power, and different sorts of operations are possible in them.

The shortcomings of associative representations were discussed in 2.2. We did not mention there that associative representations have the advantage that they *are* invertible. Indeed, the descriptive term "backchaining" aptly describes the sense in which the diagnostic process moves backwards though the associative model.

We shall preserve this property of associative systems in the models we use for hypothesis generation. Meanwhile, we will preserve the advantages of model-based reasoning by maintaining data structures that cement the models we use for hypothesis generation to our testing models.

Models must be invertible to be useful in hypothesis generation. To see this, consider the use of a computer-based numerical simulation of a suspension bridge. Given the simulator, a set of variable specifications and a series of loadings, we may

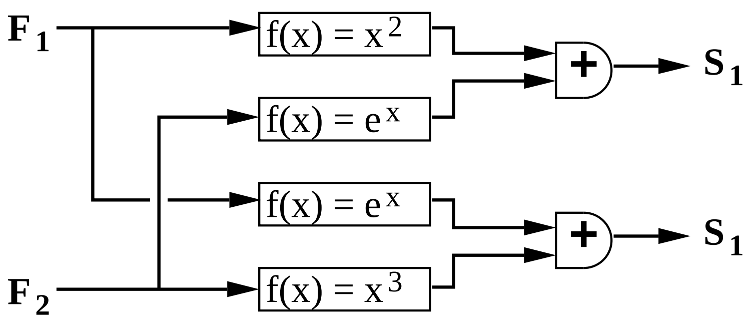


Figure 3.1: Intractable Equation Sets Emerge from Nonlinear Components

predict how the bridge twists and whether the bridge will fail. However, we cannot enter a description of a collapsed bridge into the program and then have it return the loadings and structural defects that would have produced that catastrophe. We cannot even use it to generate a set of possible loading and defects that might have caused the collapse. The model does not go from effects to causes. It is not invertible. It is not suitable for hypothesis generation.

3.4.1 Sources of Non-Invertibility

This non-invertibility is not, in general, an artifact of the way that the simulation model was written. Rather, the difficulty resides in solving the mathematics that describe the mechanism being diagnosed. For instance, consider a simple mechanism of one input F and one output S, and suppose that $S = f(F)$ where f is a polynomial of degree greater than four. Then, when going from cause to effect, we can calculate S from F directly, but must use an iterative method to go from S back to F.

There are several sources of non-invertibility in models. The most obvious is that the inverted model would be underconstrained – the component has more inputs than outputs. There are other important sources, as well.

The next source is algebraic complexity. A set of equations may give a one-to-one mapping between inputs and outputs, and yet it may be intractable to solve the equations in terms of the outputs. Consider the equations

$$\begin{array}{ll} S_1 = e^{F_1} + F_2^3 & S_1, S_2, F_1, F_2 > 0 \\ S_2 = e^{F_2} + F_1^2 & \end{array}$$

The equations are not solvable for F_1 and F_2 in closed form, and finding a series solutions will take ingenuity. Such intractable equation sets emerge from component connection models if the component behavior is not linear. These equations model the mechanism shown in Figure 3.1. If S_1 and S_2 depart from their nominal values; how can we determine what input values of F_1 and F_2 are compatible with the observed behavior?

Note that the behavior of the individual components is easily inverted. It is the emergent behavior of the mechanism which is difficult. Mechanisms with more components quickly become intractable.

This argument covers components which are modeled by algebraic constraints, or by equations. But components are not always modeled by equations alone.

3.4.2 Algorithms as a Source of Non-invertibility

Components are often modeled by algorithms. These algorithms compute the outputs of the component from its inputs. In general, they are not invertible; for instance, they may encode non-invertible approximation methods. Even if there is a one-to-one function between the component's inputs and outputs, an approximation method appropriate to go from the input to the output may be inappropriate to go the other direction. To invert the algorithm effectively, one would need some other approximation method – one not present in the algorithm. For this and other reasons, there is no general method for inverting algorithms.

It is sometimes possible to code a companion algorithm that will serve as the inverted model of a component. If we do so, we will have taken the first step towards abandoning our commitment to work with a single model, for the models will not in general be exact inverses. For instance, suppose that both algorithms are iterative approximations and we take some set of inputs, feed it to the first algorithm, and feed its outputs to the second algorithm. We will get back only an approximation of our original inputs; the models will have diverged.

But even if we can construct a inversion algorithm for each component, we have not solved our problem. The modeling of the entire mechanism is itself encoded in an algorithm, and this algorithm specifies protocols for the conditions under which the algorithms for the components are run. This algorithm itself is not, in general, invertible.

Our response to this dilemma is to abandon the attempt to invert component

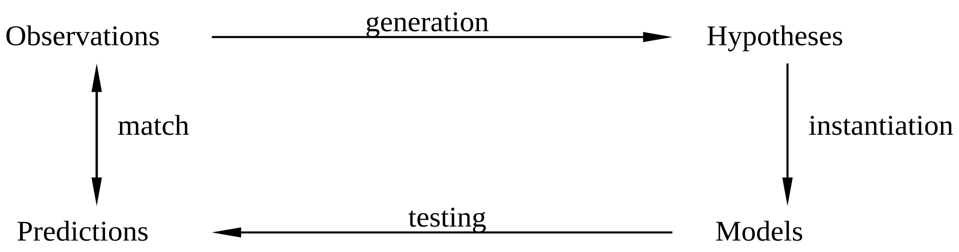


Figure 3.2: Generate-and-Test Architecture for Model-Based Diagnosis

models, or even to closely approximate the inversion of component models. Instead, we will continue to use complex, non-invertible component models for simulating component behavior, and use linear approximations to components when we need to generate hypotheses. We keep the more precise nonlinear and algorithmic models and use them as a check on our hypotheses.

All of this illustrates why the most precise models can be used for hypothesis testing, but not for hypothesis generation. Nonlinear mathematical models of mechanisms capture relationships between causes and effects that no invertible model can capture. By using invertible models, we sacrifice descriptive power in order to facilitate hypothesis formation. In model-based diagnosis, we remedy this loss by keeping the model with more descriptive power around, using it to test our hypothesis.

3.5 Generate-and-Test with Simulation Models

Model-based diagnosis (Figure 3.2) involves the use of a simulation model to test hypotheses. The hypotheses may be generated from the observed symptoms by any suitable means. The most promising hypothesis is selected as the fault hypothesis: a set of inputs to the model. The model is run and predicts a behavior. This behavior is compared to the observations—successful matches support the candidate hypothesis; failed matches contradict it.

Any method which supports reasoning from effect back to causes might be suitable for generating candidate hypotheses; this includes decision trees, first-generation associative expert systems, dependency backchaining systems, constraint suspension systems, calculations from linear models, or even a pattern match against a finite list of pre-enumerated possible faults.

To be tested, a hypothesis must be instantiated into a model, as in Figure 3.2. As we have discussed, the generation method must sometimes use a different representation than used by the testing model. Moving involves more than translation; the testing model may require information not present in the generation model.

For instance, suppose in the medical domain that we used a rule-based expert system for hypothesis generation and numerical simulation of the body for hypothesis testing. The expert system might suggest that a particular sick patient was suffering from food poisoning. To test that hypothesis, we would have to instantiate a qualitative hypothesis ("toxin in food") into a quantitative input to the simulator ("2 mg of toxin"). Too high an estimate and the testing model produces a behavior in which the patient has died; too low produces a behavior showing no ill effects. Either of these predictions fails to match the observed symptoms, even though the hypothesis is qualitatively correct, in that it correctly identifies the cause of the illness. If the generated hypothesis gives no guidance in estimating the quantity of toxin ingested, the diagnostic system may fail to confirm the (qualitatively correct) hypothesis.

Reasoning with multiple representations will oblige us to translate between them, [de Kleer, 1977]. With each shift from one representation to the other, information will either be gained or lost. In our diagnostic work, information is lost when we go from a more exact nonlinear numerical simulation to a linearized model. After we have generated a hypothesis by identifying a variable, we must add information (the variable estimate) before we can instantiate the hypothesis and run in the testing model.

When the variable value cannot be calculated exactly we face the task of *variable estimation* – computing estimated values for candidate fault variables.

3.6 Generating Hypotheses

We have said that, although there are many methods of generating fault hypotheses, different methods will not be equally adequate for instantiation into a particular simulation model. In the model-based diagnosis of nonlinear, continuous mechanisms, an adequate hypothesis will include an estimate of the magnitude of the fault. To be confirmed, the instantiated hypotheses (when it is run in the testing model) must generate a behavior that is close enough to the observed faulty behavior to match.

In this report, the hypothesis generation is broken into four stages

1. Variable identification,

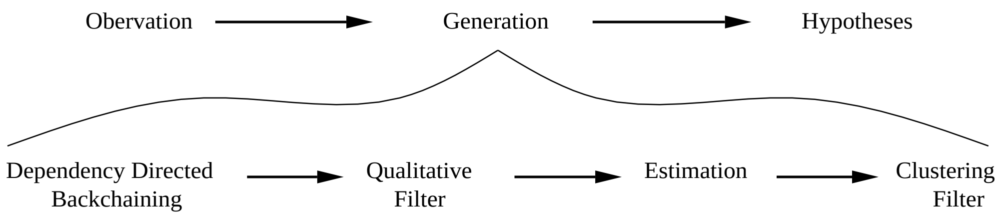


Figure 3.3: Candidate Generation Is Broken into Several Steps.

2. A qualitative filter,
3. Variable estimation, and
4. A clustering filter.

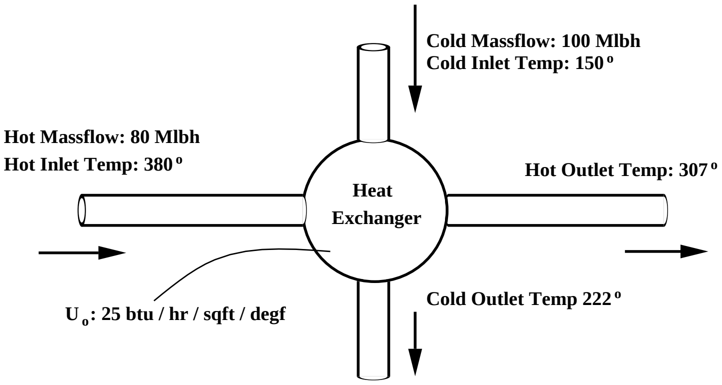
This is shown in Figure 3.3.

In each of the first three phases of the generation, a different aspect of the generating model is used. In variable identification, only the component connectivity information is used. Next, the same connections are retraced to provide the approximate numerical information that will be used in variable estimation. But before this is done, the numerical information will be abstracted into a qualitative description, $(+, 0$ and $-)$ for use in the qualitative filter.

This report uses linear approximations of components for variable estimation. This approach initially yields not just a single estimate, but instead an estimate from each symptom in the behavior. This supplies each fault hypothesis with multiple estimates of its variable value. These multiple estimates will first pass through a *qualitative filter*, eliminating some hypotheses. In the remaining hypotheses, the estimates will be arbitrated to a final estimate. Then, a *clustering filter* will eliminate more hypotheses, based upon the extent of the agreement between the multiple and final estimates. Finally, remaining instantiated hypotheses will be sent to the testing model to be tested and refined.

Variable identification uses dependency directed backchaining to identify all the candidate faults. The connections in the component model of the mechanism are followed back from the symptoms. Since this section considers only single-fault failures, only symptoms which are causally upstream from *all* the symptoms are considered. Section 3.12 will address multi-fault failures, it will consider all possible faults upstream from *some* symptom.

Section 2.3.4.5 discussed other methods of variable identification. For our work



In this simple mechanism, we consider three possible faults. Our reading of the massflows of either the hot or the cold streams may be in error, or the exchanger's heat transfer coefficient may not be at its nominal value.

Figure 3.4: A Single Component Mechanism

in diagnosis, we chose the dependency-directed backchaining method because of its similarity to the way we build the linear model of the mechanism. Both tasks call for tracing back through the mechanism, connection by connection, searching for the sites of candidate faults. In variable identification, finding the candidate faults is an end in itself; in building the linear model, equations describing each of the components along the path are gathered together to compute the gain between the candidate fault and the symptom. But the path followed is the same.

Other methods of variable identification might be used at this point, if another method for generating variable estimates is available, (see D'Ambrosio [D'Ambrosio, 1990] for a discussion of generating variable estimates using heuristics. We prefer the approach of generating linear models for two reasons. It allows us to ground our variable estimates in our knowledge of component behavior. Further, it allows us to exploit the agreement or disagreement between symptoms; we will pursue this point further in our discussion of the clustering filter.

3.6.1 Qualitative Filtering

Even when no accurate numerical model of a mechanism is available, we may have qualitative knowledge of its input/output behavior. Precisely, we may know the signs of the partial derivatives of each of the mechanism's outputs with respect to each of its inputs.

$$\frac{\partial S}{\partial F} \text{ is known.}$$

This qualitative knowledge may be more certain than the approximate numerical knowledge of the mechanism.

We have already suggested that each symptom makes a prediction about each fault variable. When there are several symptoms, there are several predictions that can agree or disagree. We will examine how those predictions can disagree numerically in Section 3.11.1. But here we will note that they can disagree qualitatively.

With each component in the mechanism, we know an approximate value for the partial derivative of each output to each input. We compose these together as we trace back the causal path from the symptom to the candidate fault. This process will be described in detail in Section 3.10. This composition yields an estimate of the a partial derivative of the value of the symptom variable S, with respect to the value of the fault variable F. The inverse of this partial derivative $\frac{\partial S}{\partial F}$, is called the *gain*; there will be a gain between each candidate fault and each symptom. (We will also refer to the ratio of output change to input change across a single component as a gain; when it is not otherwise clear from context we will distinguish these by referring to fault/symptom gains and to component gains.)

Suppose that there are m candidate faults and n symptoms.[2] Let F_i and S_j denote the values of the i^{th} fault and the j^{th} symptom, with nominal values of $\hat{F}_i$ and $\hat{S}_j$ and deviations from nominal values $\overline{F}_i$ and $\overline{S}_j$, so

$$\overline{F}_i \equiv F_i - \hat{F}_i \qquad\qquad \overline{S}_i \equiv S_i - \hat{S}_i$$

Let the estimated fault/symptom gain between $\overline{F}_i$ and $\overline{S}_j$ be G_{ij}. Dividing the magnitude of $\overline{S}_j$ by G_{ij} gives a prediction, $\overline{P}_{ij}$, of $\overline{F}_i$

$$\overline{P}_{ij} \equiv P_{ij} - \hat{F}_i = \overline{S}_j / G_{ij}$$

Using $[x]$ to indicate the sign of x, we have the qualitative relationship

$$[\overline{P}_{ij}] = [\overline{S}_j]/[G_{ij}]$$

Let $\mathbf{P}_i$ be the vector of the signs of $\overline{P}_{i1}$ through $\overline{P}_{in}$. If all of the elements of $\mathbf{P}_i$ are $+$, the symptoms unanimously predict that F_i has increased from its nominal value. If all of the elements of $\mathbf{P}_i$ are $-$, they unanimously predict that F_i has decreased. If the elements of $\mathbf{P}_i$ are mixed, then they contradict each other, and the i^{th} fault hypothesis is rejected on qualitative grounds.

[2]Notation used in this chapter is summarized in Table 3.3 on page 81 at the end of this chapter.

Fault Variable	*Observation*	
	T_{HotOut}	$T_{ColdOut}$
W_{Hot}	$+$	$+$
W_{Cold}	$-$	$-$
U_o	$+$	$-$

Table 3.1: Signs of Gains from Fault Variables to Observation Variables

3.6.2 The Single Exchanger Example – *(i)*

To make these ideas clearer, let us introduce an example – the diagnosis of a mechanism with only one component, a single heat-exchanger, shown in Figure 3.4.

For simplicity, let us assume that there are only three faults that this mechanism may suffer:

- W_{Hot}, the hot stream massflow, may have deviated from its nominal value of 80 Mlbh.[3]
- W_{Cold}, the cold stream massflow, may have deviated from its nominal value of 100 Mlbh.
- U_o, the overall heat transfer coefficient, may have deviated from its design value of 25 btu/hr/ft^2/°F.

There are only two output measurements made of the system: T_{HotOut} and $T_{ColdOut}$, the outlet temperatures for the two streams.

Suppose that a fault occurs. W_{Hot} drops from its nominal value of 80 Mlbh to 65 Mlbh. The symptoms will be that both T_{HotOut} and $T_{ColdOut}$ will drop.

All three of the possible faults are causally upstream from both symptoms; all of them will be evoked as candidate faults by dependency-directed backtracking. But even a small amount of qualitative knowledge is enough to eliminate one of the fault hypothesis.

[3]All temperature data is in degrees Fahrenheit (°F). All massflow rates are in thousands of lbs/hr (Mlbh). All surface area is in ft^2. All heat transfer coefficients are in btu/hr/ft^2/°F. Physical property data for hydrocarbons from Perry, [Green, 1984].

Without any numerical knowledge of the mechanism, it is still possible to suggest the qualitative response of the outputs to any of the inputs. Table 3.1 shows the qualitative correlation of faults to symptoms. Both output variables have dropped; this is consistent with either a positive change in W_{Cold} or a negative change in W_{Hot}.

But no change in U_o is consistent with both symptoms. decreased T_{HotOut} is inconsistent with a decreased U_o and decreased $T_{ColdOut}$ is inconsistent with an increased U_o . We can eliminate U_o as the fault variable, without doing any numerical analysis. This leaves us with two fault hypotheses about two candidate fault variables: W_{Hot} and W_{Cold}.

3.7 Variable Estimates & Linear Models

When diagnosing continuous mechanisms, we prefer that our generation model supply us with both fault variable identifications and fault variable estimates.

In this report, we use linear approximations of components to build linear models of mechanisms. The linear model is a set of linear simultaneous equations.

It might be thought that we could just build and solve such a model directly, skipping the work of variable identification and qualitative filtering. But this approach has several problems.

First, the numbers of inputs and outputs may not match. If there are more inputs than outputs, there may either be unmeasured outputs or the mechanism's inputs may be underconstrained. If there are more outputs than inputs, some of the outputs are redundant. However, in the presence of the inaccuracies in linearizing the components, the linear model will be overconstrained and have no solution.

Second, even if the number of inputs and outputs match, the solution generated may not be consistent with the single-fault assumption. That is, a solution to the linear constraints is likely to show that the best fit to the observed behavior is given by a hypothesis in which *all* of the input variables vary from their nominal values. We would prefer, however, to generate a hypothesis in which only one input variable varies from its nominal value. We prefer this even though there is no value for that variable which will give an exact fit to the observed behavior (in the linear model.)

Third, solving the entire linear constraint set does not separate those possible faults which are causally linked to the symptoms from those which have no such linkage.

We therefore build our linear models only after variable identification. The fault variable is an unknown, and a causal path between the fault and the symptom is one equation. In our implementation, each feedback path adds another equation and another unknown. These equations are solved to give the linear function.

We will discuss our reasons for using linear models further in Section 3.9. But first we give our definition of the estimate of variable, and then begin an example to show how it is used.

In Section 3.6.1 we showed how the value of a symptom $\overline{S}_j$ could be used to make a prediction $\overline{P}_{ij}$ of a fault variable $\overline{F}_i$. We define the estimate, E, of a parameter as the mean of the predictions:

$$\overline{E}_i \equiv E_i - \hat{F}_i = \frac{\sum_{j=1}^{n} \overline{P}_{ij}}{n}$$

3.7.1 The Single Exchanger Example – *(ii)*

We return to the single heat-exchanger example that we introduced in Section 3.6.2.

When W_{Hot} drops by 15 Mlbh, T_{HotOut} drops from 307.0° to 281.0°, (Δ of $-14.0°$) and $T_{ColdOut}$ drops from 221.8° to 220.2°, (Δ of $-1.6°$.) If we observe these drops, how may we determine which fault is the cause?

The response of T_{HotOut} and $T_{ColdOut}$ to the three faults is shown graphically in Figure 3.5. These relations were generated using the iterative method outlined in Section 1.3.2.

We may approximate these functions with linear functions. The dotted lines in the graphs in the figure show such approximations; in this example, we use the tangents to the functions at the nominal values. We have two pairs of approximating functions – one pair for each remaining possible fault.

We can use these linear functions to predict the values of candidate fault variables. Table 3.2 shows the predictions for the example. The gains in the table are the slopes of dotted lines in Figure 3.5. Let us work through one of the calculations. T_{HotOut} has changed by $-14.0°$. W_{Hot} has a nominal value of 80 Mlbh. There is a gain of .52

Two observable variables' response to deviation in the value of three candidate fault variables in the single heat-exchanger mechanism shown in Figure 3.4.

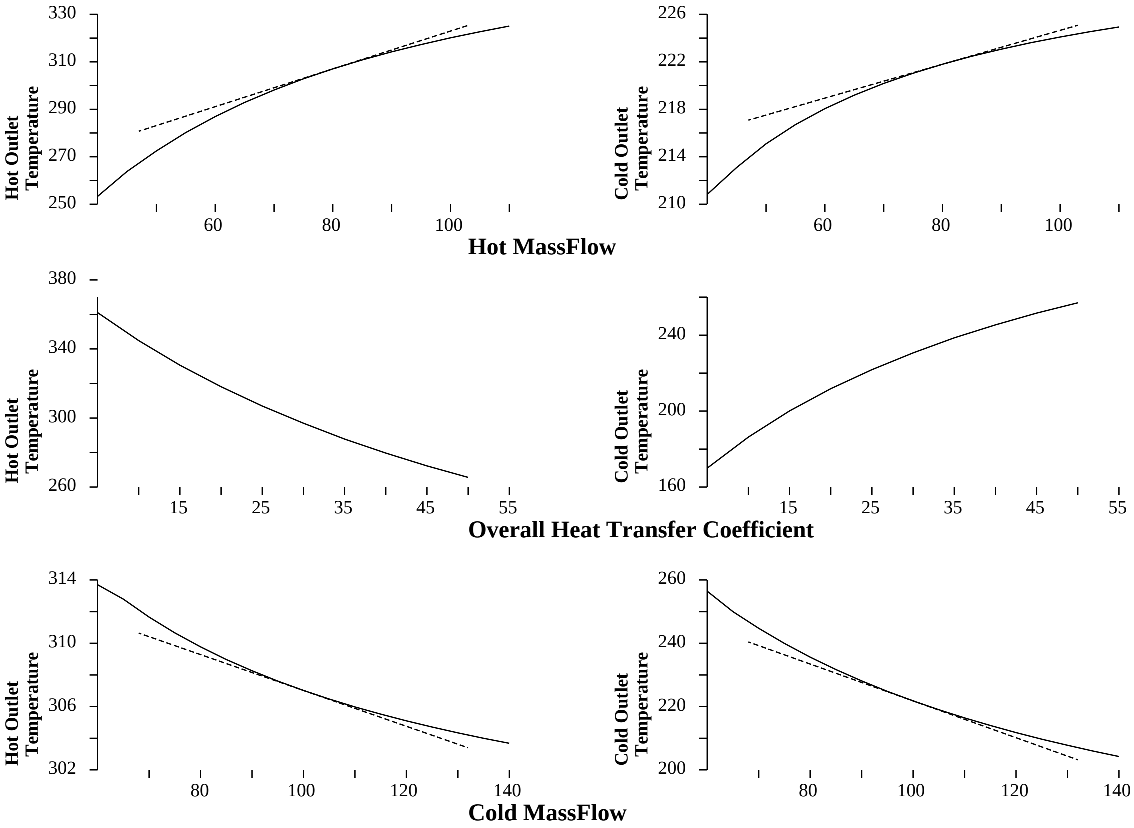


Figure 3.5: Nonlinear Response to Changes in Three Input Variables

Observation	*Candidate Fault Variable* Hot Massflow Nominal Value: 80 Mlbh	 Cold Massflow Nominal Value: 100 Mlbh
Hot Stream Outlet Temperature Nominal Value: 307.0° Observed Value: 281.0° $\Delta = -14.0°$	Gain: .52 $\frac{°F}{\text{Mlbh}}$ Predicted Fault Value: 53.1 Mlbh	Gain: 0.093 $\frac{°F}{\text{Mlbh}}$ Predicted Fault Value: 62.8 Mlbh
	Estimate: 57.9 Mlbh	
Cold Stream Outlet Temperature Nominal Value: 221.8° Observed Value: 220.2° $\Delta = -1.6°$	Gain: -0.17 $\frac{°\text{F}}{\text{Mlbh}}$ Predicted Fault Value: 182 Mlbh	Gain: -0.56 $\frac{°\text{F}}{\text{Mlbh}}$ Predicted Fault Value: 103 Mlbh
	Estimate: 142.5 Mlbh	

Table 3.2: Prediction of Fault Variable Values for a Single Exchanger

$\frac{°F}{\text{Mlbh}}$ between them. The value of W_{Hot} predicted by T_{HotOut} is then

$$80 \text{ Mlbh} + (-14.0°)/(.52\tfrac{°\text{F}}{\text{Mlbh}}) = 53.1 \text{ Mlbh}.$$

The other predictions are calculated in the same way. The estimate for a fault variable is the mean of the predictions. For the hot stream outlet temperature, this was

$$\frac{53.1 \text{ Mlbh} + 62.8 \text{ Mlbh}}{2} = 57.9 \text{ Mlbh}.$$

The table shows that each hypothesis gets an estimate of its fault variable's value; it uses the two predictions about its fault variable. The two predictions for the W_{Hot} hypothesis are that its temperature has dropped to 53.1 and 62.8 Mlbh, respectively, with a mean of 57.9 Mlbh. The two predictions for the W_{Cold} hypothesis are that its temperature has increased to 182 Mlbh and 103 Mlbh, respectively, with a mean of 142.5 Mlbh.

The mean of the individual predictions is our estimate of the fault variable value. Before this calculation, we believed that the fault was either caused by low hotstream massflow or high coldstream massflow. We now have numerical estimates for each of these hypotheses. We have not yet said which one was more plausible.

We will develop this example further in Section 3.11.

3.8 Building Linear Models from Component Descriptions

In model-based reasoning, we seek to generate descriptions of mechanism behavior by composing known component behaviors. We here give a procedure for building an approximate linear model of a mechanism, using approximate linear models of the components and knowledge of the connectivity of the components in the mechanism.

We must first obtain the linear component models. We anticipate that operating conditions will change frequently, and that the nominal values of many of the dependent variables of the model will change with them. As these values change, so will the component gains. We have, then, a choice of approaches,

1. Using some fixed value for the component gains, and ignoring any changes that result from changing operating conditions, or
2. Calculating or estimating all the component gains each time the operating system changes and storing them, or
3. Calculating them or estimating them when they are needed.

For each component we have available a set of gains, relating the inputs to the component and the outputs. These are encoded in some form; we will choose to call our encoding a *rule*, because we will be backchaining from gain to gain in a fashion similar to the backchaining done in rule-based systems. These rules are of the form

$$(V_i \; V_j \; \Gamma_{ij})$$

meaning that

$$\frac{\partial V_j}{\partial V_i} = \Gamma_{ij}$$

where $V_i, \; V_j \in \mathcal{V}$, $\mathcal{V}$ is the set of component variables and Γ_{ij} is the component gain In practice, each V is encoded as a (component-name, variable-name) pair, leading to an overall encoding as

```
((component1 variable1)(component2 variable2) gain)
```

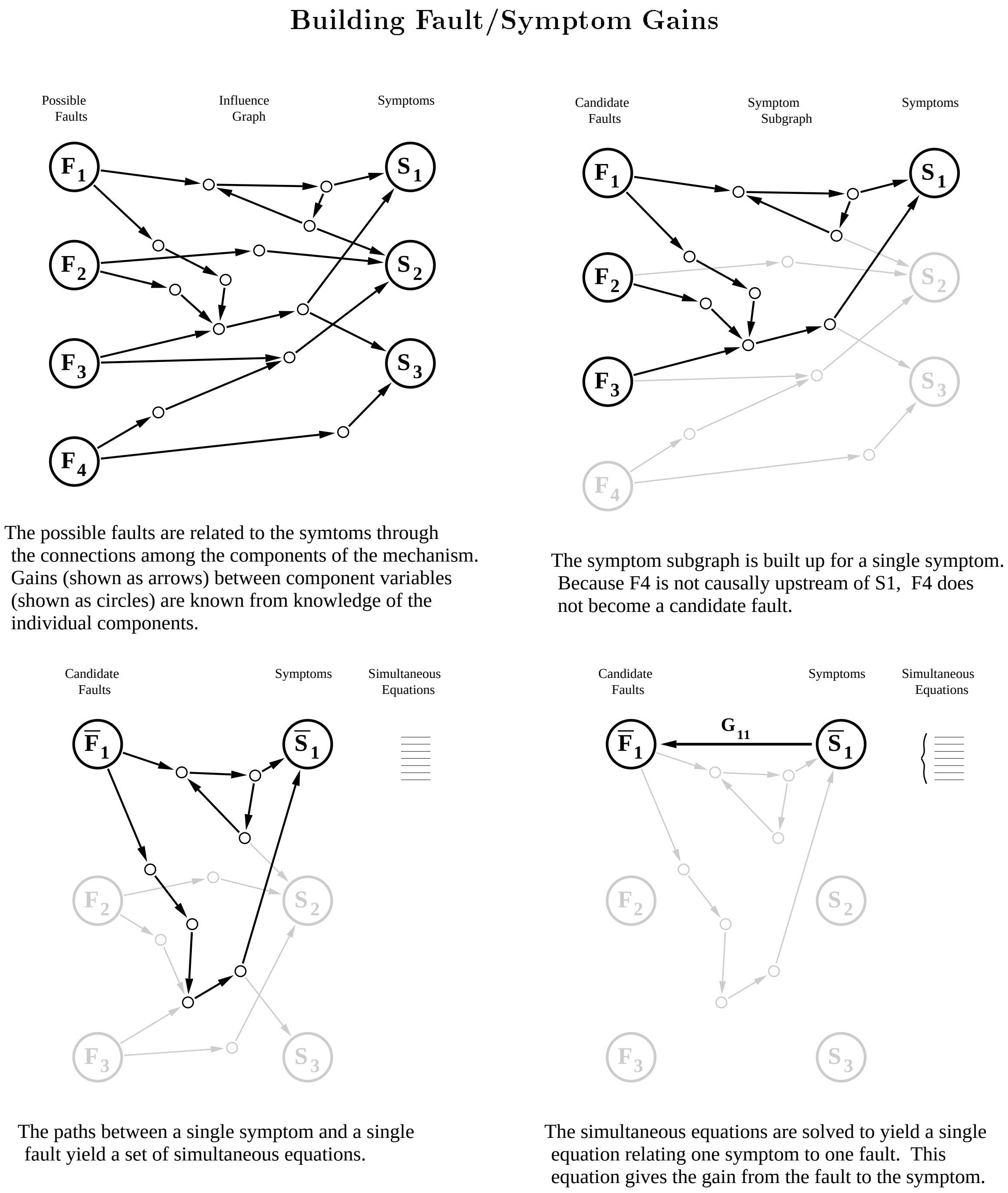


Figure 3.6: Linear gains are built up from knowledge of components.

Single Fault Diagnosis

The panels in this figure continue a series that began in Figure 3.6 on page 60.

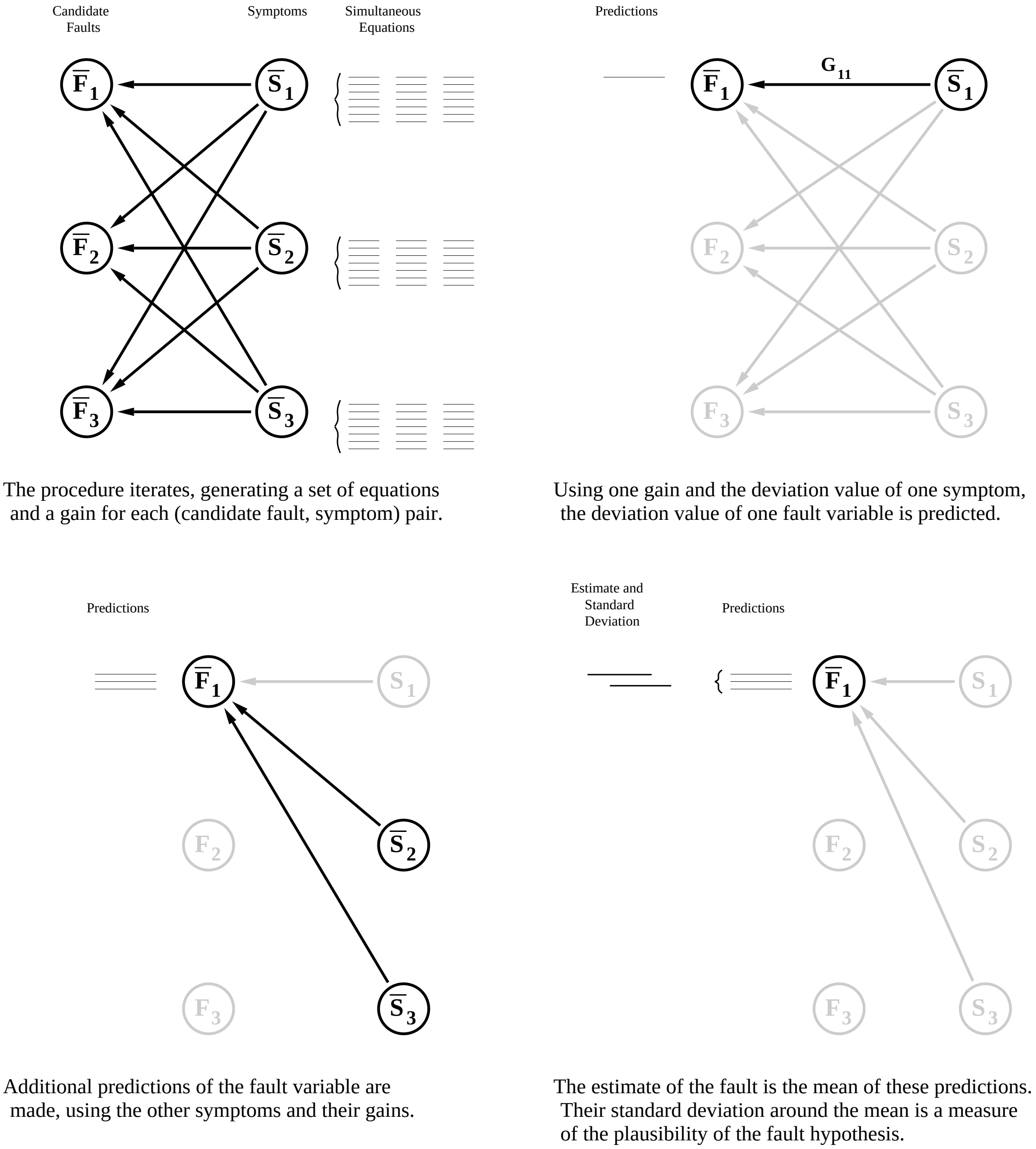
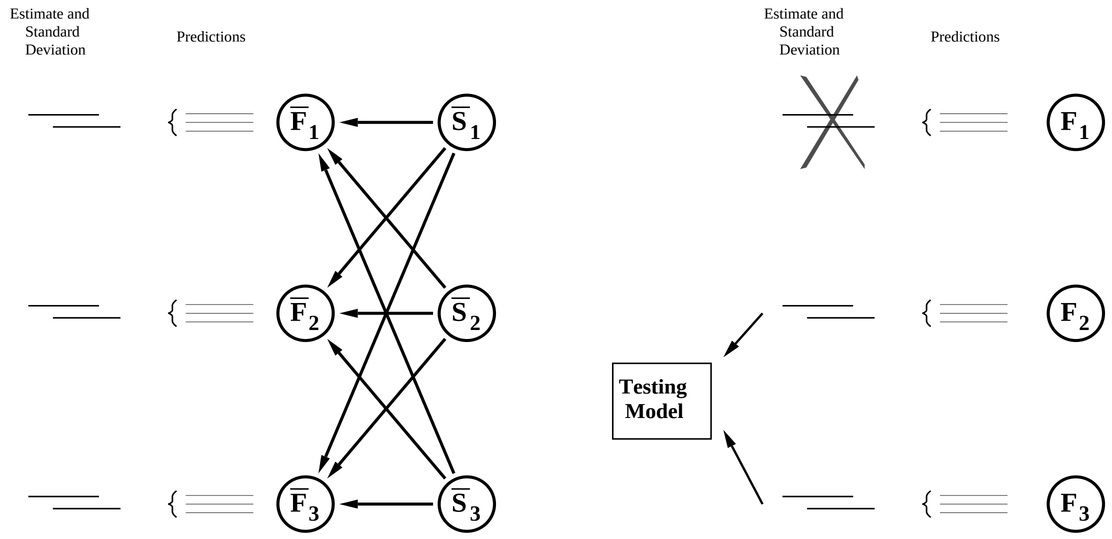


Figure 3.7: Gains are used to estimate values for candidate faults.

Single Fault Diagnosis

The panels in this figure continue a series that began in Figure 3.6 on page 60, and continued in Figure 3.7 on page 61.



The entire procedure is repeated for each remaining candidate fault, making an estimate of its value and obtaining a clustering score from its standard deviation.

The clustering scores are judged. A hypothesis with poor clustering is rejected, while hypotheses with good scores are tested by the testing model.

Figure 3.8: The estimate and standard deviation is calculated for all candidate faults.

(with the understanding that the gain may actually be computed as needed, rather than stored.) When describing the I/O behavior of a component, *component1* will be the same as *component2*. An example (with *HX* indicating 'heat-exchanger' and 'hstr' indicating 'hot-stream')

```
((HX-12 hstr-mass-inflow) (HX-12 hstr-outlet-temp)    3 °F/Mlbh).
```

There will also be rules that encode the connectivity of the network. These will typically have a gain of 1, as the quantity is just being passed from one side of a port to the other –

```
((HX-12 hstr-outlet-temp)(pipe-1 inlet-temp) 1).
```

A directed *influence graph* can be built from these rules in which each V is a node and which each rule $(V_a\ V_b\ \Gamma_{ab})$ is a directed arc from V_a to V_b and labeled by Γ_{ab}. An influence graph is shown in the first panel of Figure 3.6. This graph may be cyclic, (but may have no self arcs, i.e., $V_a \neq V_b$.) Root nodes in this graph (nodes with no incoming arcs) will either be possible fault variables or independent parameters of the system which can be assumed never to vary (the variables that parameterize the dependency of a liquid's viscosity on its temperature would be an example); call the set of these unvarying independent parameters $\mathcal{V}_p$. Leaf nodes are normally observable variables, (a leaf node that was not observable would represent a V that was not seen and did not influence any other V, and thus superfluous.) However, observable variables need not be leaves; we may have sensors measuring variables that are causally upstream of other variables.

Notice, for all symptoms, that $S \in \mathcal{V}$. We now select, in turn, for each symptom S_j, a *symptom subgraph* – a subgraph, T_j, of the influence graph which contains only those nodes causally upstream of that symptom. A symptom subgraph is shown in the second panel of Figure 3.6.

We construct this by coloring S_j, then coloring any node uncolored V_u for which there is an arc $(V_u\ V_c\ \Gamma_{uc})$ such that V_c is colored. We iterate on this coloring step until no more nodes may be colored; the desired subgraph is the colored nodes and the arcs between them. Note that this subgraph may still contain cycles.

Let $\mathcal{F}$ be the set of all possible component faults, $\mathcal{F} \subset \mathcal{V}$. We generate the set of candidate faults, $\mathcal{F}'$ by selecting the m faults, F_1 through F_m which are found in all of the symptom subgraphs:

$$\mathcal{F}' = \{F \in \mathcal{F} | \forall j\ (0 \leq j \leq n)\ F \in T_j\}$$

For a candidate fault F_i, we generate the gain G_{ij} to S_j. We do this by asserting and solving a large set of simultaneous linear equations. Let there be l nodes in the subgraph, (this includes S_j, all $F \in \mathcal{F}'$, and all nodes representing fixed parameters, $\mathcal{V}_p$.) Let the predecessors of a node V_b be nodes V_q through V_r. For each node in T_j that has an incoming arc, we assert an equation indicating that the deviation in that node is the sum of the products of the deviation in node's predecessors and the gain along the arc. That is, for each V_b *except* F_i, assert:

$$\overline{V}_b = \sum_{x=q}^{r} \overline{V}_q \cdot \Gamma_{xb}$$

For those V in T_j (except F_i) that have no incoming arcs, this sum will be zero. We are calculating $\frac{\partial S_j}{\partial F_i}$ and are therefore holding the deviation in all F_j $j \neq i$ to zero; we are also keeping the deviation in the nodes in $\mathcal{V}_p$ at zero.

The only rules that will contribute to G_{ij} will lie along the path or paths between F_i and S_j. Holding all but one of the inputs constant to the symptom subgraph removes all other rules algebraically; the terms that those other rules contribute drop out and we are left only with the path between F_i and S_j. This is shown in the third panel of Figure 3.6.

For the l nodes in the subgraph, we have asserted $l - 1$ linear equations with l different deviation variables. Notice that none of these equations have constant terms. We use the usual algebraic methods to reduce this to one linear equation in two deviation variables, $\overline{S}_j$ and $\overline{F}_i$ and a constant, which is G_{ij}

$$\overline{F}_i \cdot G_{ij} = \overline{S}_j.$$

This is shown diagrammatically in the fourth panel of Figure 3.6.

We repeat this procedure for each $F_i \in T_j$; this will give us G_{1j} through G_{mj}. We then repeat this entire process for each of the symptoms, using the symptom's subgraph, until we have built up a gain for every fault/symptom pair. This is shown diagrammatically in the first panel of Figure 3.7.

This completes our linear model. Starting with linear models of components, we have built up a linear model of the entire mechanism, in a form in which the components have been abstracted out, their behaviors aggregated into the behavior of the whole mechanism. In this aggregation, we have abstracted away the loops, the feedback and the reconvergent fanout that defeat local reasoning methods. In their place, we have an approximate model which will take symptoms as inputs, and will give estimates of faults as outputs.

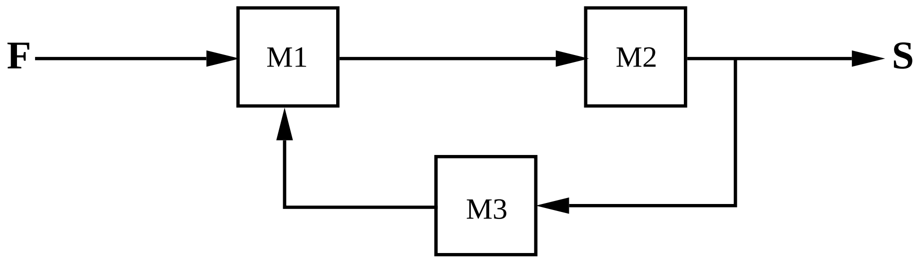


Figure 3.9: A Mechanism with Three Components and Feedback

3.9 Composition and Invertibility

It may be argued that, although it is computationally intensive to build up the fault/symptom functions from the constraints, it is not infeasible. Once built, many functions, such as quadratic functions or piecewise linear functions can be fit to them. These functions are easily invertible. Why insist that the approximating functions that describe the components be linear? Why not use a better approximating function, like piecewise linear or quadratic?

Such an approach succeeds when applied to a single component. It may also succeed when the components of a mechanism are joined in a tree structure, so that there is exactly one path between any candidate fault and each symptom.

Even small mechanisms involving feedback will defeat this approach. Figure 3.9 shows a three-component mechanism. In it, the deviation $\overline{S}$ in the output variable S depends upon the deviation $\overline{F}$ of input F as

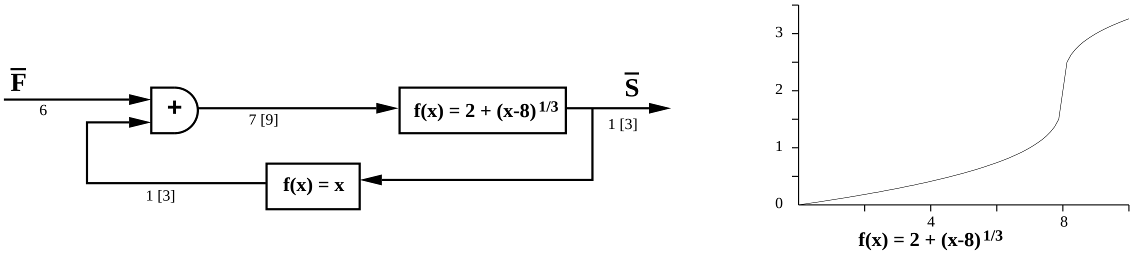
$$\overline{S} = M2(M1(M3(\overline{S}), \overline{F}))$$

where $M1$, $M2$ and $M3$ are the input/output functions of the three components. (Again, the overline indicates deviation values.) Notice that the relation gives $\overline{S}$ in terms of $\overline{F}$ and $\overline{S}$ itself. We cannot solve for $\overline{S}$ as a function of $\overline{F}$ alone and in terms of $M1$, $M2$ and $M3$ and their inverse functions. Indeed, even though $M1$, $M2$ and $M3$ are invertible, their composition may not be functional – there may be multiple values of $\overline{S}$ consistent with a given $\overline{F}$. For instance, if just one of the functions is nonlinear

$$M1(x, y) = (x + y) \qquad M2(x) = x, \qquad M3(x) = 2 + (x - 8)^{1/3}$$

then for $\overline{F} = 1$, the system has two stable outputs of $\overline{S} = 1$ and $\overline{S} = 3$, ($\overline{S} = 2$ satisfies the functions but is unstable.) This is shown in Figure 3.10.[4]

[4]This model of an example mechanism suggests an interesting problem for model-based diagnosis.



For input of $\overline{F} = 6$, this network has two stable outputs, $\overline{S}$, of 1 and 3. (Internal values in the networks for the output of 3 are shown in brackets.) The nonlinear function is shown graphically at right.

Figure 3.10: Nonlinear Feedback Loop with Two Stable Outputs

However, if the components are represented by linear functions

$$M1(x,y) = m_1 \cdot (x+y), \qquad M2(x) = m_2 \cdot x, \qquad M3(x) = m_3 \cdot x,$$

with m_2, m_1, m_3 constant, then

$$\overline{S} = m_2 m_1 (m_3 \overline{S} + \overline{F}),$$

which can be solved for $\overline{S}$

$$\overline{S} = \frac{m_1 m_2 \overline{F}}{1 - m_1 m_2 m_3}.$$

The lesson here is clear. To do hypothesis generation in model-based reasoning, it is not enough to represent the behavior of components by functions which are invertible. The functions yielded by composing the component functions must also be invertible. In this report, in which the mechanisms contain feedback, this will limit us to linear models of components for hypothesis generation. We shall briefly discuss nonlinear extensions in Section 5.1.

There is a related mechanism in which $M3(x) = 1 + (x-1)^{1/3}$. This mechanism has the nominal behavior in which $\overline{F}$, $\overline{S} = 0$. It also has an unexpected behavior in which $\overline{F} = 0$, $\overline{S} = 2$. In Section 2.3.1, we said that, in model-based diagnosis, if the model is correct, all the discrepancies between observation and prediction arise from (and can be traced back to) defects in the device. We further said that each fault in the mechanism corresponds to, and is localizable to, some fault in an individual component. This does not hold for the model of this mechanism. The model accurately captures the behavior of the device and predicts the nominal behavior. The device exhibits a faulty behavior. The model is capable of confirming that the device can exhibit this faulty behavior, and yet the fault is not localizable to a defect in any component.

3.10 Gains

When faults and symptoms have magnitudes, and when symptoms can be linked back to a candidate fault by a series of invertible quantitative constraints, then each symptom makes a prediction about the magnitude of that fault. The use of a symptom and a gain to predict a fault is shown diagrammatically in the second panel of Figure 3.7.

Suppose that we hypothesize that a particular fault has caused a symptom. If all the input variables (except the fault variable) are held constant at their nominal values, then the symptom variable is a function of the fault variable. We again use overlines to indicate deviation variables for the fault F and the symptom S.

$$\overline{S} = f(\overline{F}) \qquad\qquad f(0) = 0$$

If f is a monotonic function in the range we are considering, it will have an inverse function which allows us to calculate the value of the fault from the value of the symptom.

$$\overline{F} = f^{-1}(\overline{S}) \qquad\qquad f^{-1}(0) = 0$$

We take the term *gain*, G from engineering parlance to denote the ratio of the deviations of the symptom and the fault:

$$G = \frac{\overline{S}}{\overline{F}}$$

If f is a linear function, G will be constant as $\overline{F}$ changes. Even if f is not linear, we may construct a linear approximation g to f^{-1}

$$\overline{F} \approx g(\overline{S})$$

where

$$g(x) \equiv \frac{x}{g^*}$$

and g^* is a constant chosen to make g a workable approximation.

We have approximated the inverse of f by g. We have a particular symptom and a fault that we have hypothesized is causing that symptom. The g function allows us to predict an approximate value for the fault variable.

In the example in Section 3.7.1 we estimated each gain of a simple mechanism by constructing a function f linking the fault and symptom variables as $S = f(F)$ and then taking the gain as the partial of the inverse function, $\frac{\partial}{\partial S}f^{-1}$. In larger mechanisms, we shall not construct the functions that link symptoms to faults. Rather, we shall approximate the gains by composing estimates of the partials of the component functions as we trace through the component connection description of the mechanism.

This prediction of the fault variable will be approximately correct *if* the fault hypothesis was right; that is, if we have correctly identified which fault variable is causing the symptom, (and if the approximation is not badly off.) If we have erred in identification, the approximation that $\overline{F} \approx g(\overline{S})$ will be completely off, as $g(\overline{S})$ will have some nonzero value, but $\overline{F}$ will have its nominal value – zero.

3.11 Clustering Guides Hypothesis Selection

Each symptom makes a prediction. That is, each symptom provides a way of predicting the value of a candidate fault variable. When there are multiple symptoms, there will be multiple predictions of each fault variable. The predictions may match well, or poorly, or not at all. The agreement between these multiple predictions is a measure of the plausibility of the fault hypothesis. Gathering multiple predictions of a fault, using them to make an estimate of the fault, and comparing their agreement as a measure of the fault's plausibility are shown diagrammatically in the last two panels of Figure 3.7.

Predictions agree by clustering tightly together around a single estimated value. They disagree by failing to cluster, by being widely dispersed.

When several hypotheses emerge from the qualitative filter, we can discriminate between them on the basis of their plausibility by examining, for each one, the tendency of the multiple predictions to cluster together. This discrimination is the basis of the clustering filter.

3.11.1 The Clustering Filter

For each hypothesis, the clustering filter generates multiple predictions of the fault variable's value. From these predictions, it calculates a single estimate of the value. It then measures how tightly the multiple predictions cluster around the estimate, and ranks the hypotheses by this measure, rejecting those hypotheses for which the clustering is to poor. Chapter 4 will report how well the clustering filter works on a large set of problems. In this section, we present a description of the *way* in which it works, a suggestive argument of why it *should* work, and a simple example showing how it *does* work. This section develops the application of the clustering filter to single-fault failures; Section 3.12 develops the extension to multi-fault problems.

Why should the clustering filter work? The clustering filter discriminates based on a score of how closely the different predictions (each based on a symptom) agree. If the mechanism were linear, the predictions would be identical and the agreement would be exact. Of course, we have chosen a domain in which most of the component functions are *not* linear. Still, many of them are "roughly" linear and the predictions stay in rough agreement.

Conceptually, the filter reflects an insight about the nature of cause and effect in a continuous domain. Often, several input variables will affect the same output variables. But each one will affect the output variables in different amounts: each fault will have a different "profile" in the output variables. When the mechanism is not linear, this profile will depend upon the magnitude of the fault. But the character of the profile will change slowly for a particular fault in comparison to the difference between the profiles of different faults. The more symptoms in the behavior, the more detail there is in the profile, and the more strongly different profiles are distinguished.

Of course, for any of this to work, there must be at least two symptoms. (A variant on this approach, which uses the connection between the fault variable and observed variables that have *not* left their nominal values, would allow this method to be applied to single-symptom faults. We have not implemented this insight, however.)

How do we measure the "tightness" of clustering? Clearly, if one pair of estimates is 12°F apart and a second pair is 30°F apart, the first pair clusters more closely than the second. But if the the two candidate fault variables have different units, no such direct comparison is possible. If a pair of symptoms could either be caused by a non-nominal temperature or a non-nominal massflow, we will be comparing a cluster of temperatures to a cluster of massflows. Which is a worse disagreement, 20°F, or 15 Mlbh?

We need to put such comparisons on a common footing; we do this by dividing by a quantity of the same dimensions to make the clustering a dimensionless quantity. We can do this either of two ways:

- by dividing through by a standard amount for each type of observation (by unilaterally declaring that, for instance, spreads of 10°F for temperatures, 10 Mlbh for massflows and 5 btu/hr/ft^2/°F for heat-transfer-coefficients shall all be treated as equal), or

- by dividing through by the distance between the averaged predictions and the nominal value.

Either way puts the differences in prediction into dimensionless form, and allows us to compare the clustering scores of different variables. We call this quantity the *observed dimensionless error*.

We define the *clustering score* of a variable to be the standard deviation of the observed dimensionless error. More formally, and continuing the notation of 3.6.1, the clustering score C for the i^{th} candidate fault variable F_i (with nominal value of $\hat{F}_i$ and mean estimate E_i) is

$$C(F_i) = \frac{\sum_{j=1}^{n}(\frac{\overline{P}_{ij}-\overline{E}_i}{\hat{F}_i-E_i})^2}{n}$$

Notice that $\hat{F}_i \neq E_i$ because the qualitative filter assures that the set of predictions does not span the nominal value.

The scoring of multiple hypotheses in the clustering filter is shown in the first panel of Figure 3.8.

The clustering filter is judging hypotheses from a different standpoint than is the qualitative filter; the clustering filter does not obviate the need for the qualitative one. To see this, suppose that two observation variables, S_1 and S_2, are causally linked to a candidate fault variable F as

$$S_1 = M^+(F) \qquad\qquad S_2 = M^-(F)$$

and that both $M^-(F)$ and $M^+(F)$ are very steep, so that a small change in F produces large changes in both S_1 and S_2.

If both S_1 and S_2 decrease by small amounts, they will give two predictions: S_1 will predict that F has *decreased* by a tiny amount and S_2 will predict that F has *increased* by a tiny amount. These two predictions will cluster very tightly together – however, the two predictions will span the nominal value for F. The qualitative filter rejects such a hypothesis, as it should. The clustering filter would not, because the two predictions *are* very close together.

Thus, we see that, even with the clustering filter in place, the qualitative filter is still necessary. Additionally, the qualitative filter is less computationally intensive than the clustering filter, and we try always to use a cheap, quick test when we can.

The second panel of Figure 3.8 shows a hypothesis being rejected by the clustering filter and other hypotheses passing and being sent to the testing model. Hypotheses rejected by the clustering filter are not discarded. Rather, the hypotheses are divided into priority classes: groups of hypotheses having roughly equal clustering scores. In the eventuality that all the hypotheses which pass the clustering filter are rejected by the testing model, the hypotheses in the highest untested priority class are still available to be reconsidered.

3.11.2 The Single Exchanger Example – *(iii)*

We continue with the ongoing example, which we last saw in Section 3.7.1 and saw summarized in table 3.2. The outcome of the exercise shows which hypothesis is correct. The two symptoms have made two predictions for The hypothesis that the observed temperature drop was caused by a drop in W_{Hot}. The cold stream symptom predicted that W_{Hot} has dropped from its nominal value of 80 Mlbh to 53.1 Mlbh; the hot stream symptom predicted a value of 62.8 Mlbh. This is not perfect agreement, but it is reasonably close.

The hypothesis that the observed temperature drop is caused by an increase in W_{Cold} also makes two predictions – that W_{Cold} has increased (from its nominal value of 100 Mlbh) to 182 Mlbh and 103 Mlbh, respectively. These predictions are extremely far apart.

Calculating the clustering scores, for the W_{Hot} hypothesis gives

$$\frac{(\frac{53.1-57.9}{80-57.9})^2 + (\frac{62.8-57.9}{80-57.9})^2}{2} = .048.$$

A similar calculation for the W_{Cold} hypothesis yields a clustering score of 0.91. The (correct) identification of W_{Hot} as the fault variable is plausible; the identification of W_{Cold} is not.

3.12 Multi-Fault Diagnosis

Some severe difficulties arise in applying any of the standard methods for fault variable identification to model-based diagnosis of continuous, nonlinear systems that include feedback.

Chapter 2 examined seven methods of variable identification: fault dictionaries, heuristic identification, neural nets, dependency backchaining, constraint suspension, incremental construction and generalized set covering methods.

Section 2.3.4.5 discussed the shortcomings in fault dictionaries and heuristic identification which motivated a model-based approach; it also made a similar criticism of neural nets. Section 2.3.4 discussed a problem arising when applying dependency backchaining methods or constraint suspension to problems with feedback or reconvergent fanout – almost any discrepancy can be accounted for by the failure of any component. In single fault diagnosis, this manifests as a lack of discrimination – the methods do not greatly prune the space of diagnoses. In multi-fault diagnosis, this manifests as a lack of any guidance in picking a second fault for the hypothesis; if one fault already covers the symptoms, what's left to guide the selection?

Incremental construction has already been used for diagnosis in continuous domains. Section 2.3.4 discussed some criticisms that have been made: it assumes that manifestations are additive, and may miss diagnoses. In our domain, faults *do* interact. For these reasons, we have not implemented an incremental construction method for multi-fault diagnosis in HEATX.

This leaves the generalized set-covering methods, which were discussed in Section 2.3.4.4. Set-covering starts by assuming that all possible faults have occurred, and then eliminating faults until an inconsistency appears. This approach assumes the availability of a theorem prover, *TP* (or some other kind of consistency checker) which, applied to the model, can decide if it is consistent to believe that a particular set of faults is the only thing wrong with the mechanism. Consistency checking is, in general, NP-complete, but in some restricted domains it is tractable. Set covering can use variations on other generation methods as the *TP*, such as constraint suspension using a TMS. A series of several tests for consistency can be used, running cheap tests early and more expensive tests later. Still, *TP* will be called on many non-minimal coverings which are consistent; the most expensive test will be called in each of these.

In this report, our model is a numerical simulation. Using it as a theorem prover on a non-minimal covering would involve giving it a set of symptoms and a covering set (the list of candidate fault parameters) then asking it to find a set of values for the candidates that generate a behavior matching the symptoms. This will be costly when the covering set is consistent even if the system finds, on the first try, a set of fault variable values that generate the symptoms. If the covering is inconsistent, the hypothesis may be screened by some cheaper test (such as the qualitative filter, or mass balances.) But when it is not, we will wait a *long* time before the simulator reports that it can make no further progress in matching the behavior, especially when the set of variables free to vary (the covering set) is large.

For this reason, we have not implemented a generalized set-covering routine for HEATX. We consider variable identification for multiple faults in continuous, non-linear mechanisms with feedback to be an open problem which this report does not solve. However, we have extended our methods of the clustering filter to multi-fault hypotheses.

3.12.1 Extending the Clustering Filter

The clustering filter can be extended to multiple faults. Section 3.8 showed, for each fault F_i and symptom S_j, how to calculate a gain G_{ij} such that

$$\partial F_i \cdot G_{ij} \approx \partial S_j.$$

Under the linearity and single fault assumptions this became

$$\overline{F}_i \cdot G_{ij} \approx \overline{S}_j.$$

We showed how to used this gain to make a prediction P_{ij} such that

$$\overline{P}_i \cdot G_{ij} = \overline{S}_j \qquad\qquad \overline{P}_i \approx \overline{F}_i$$

and how to combine these predictions into an estimate E_i

$$\overline{E}_i = \frac{\sum_{j=1}^{n} \overline{P}_{ij}}{n} \qquad\qquad \overline{E}_i \approx \overline{F}_i.$$

In the single fault case, there was one hypothesis for each candidate fault, and we did not need a sharp distinction between them. For multiple faults, hypotheses will be sets of fault variables, and individual fault variables may be members of several hypotheses.

In multi-fault hypotheses, the sum of the effect of the multiple faults cause each symptom. A multifault hypothesis H_k holds that

$$\sum_{i=1}^{n} \overline{E}_i \cdot G_{ij} \cdot Mem(i,k) = S_j$$

where $Mem(i,k)$ is the membership function

$$\begin{aligned} Mem(i,k) &= 1 \quad \text{if } F_i \in H_k, \\ Mem(i,k) &= 0 \quad \text{otherwise.} \end{aligned}$$

Multiple Fault Diagnosis

The panels in this figure continue a development that began in Figure 3.6 on page 60.

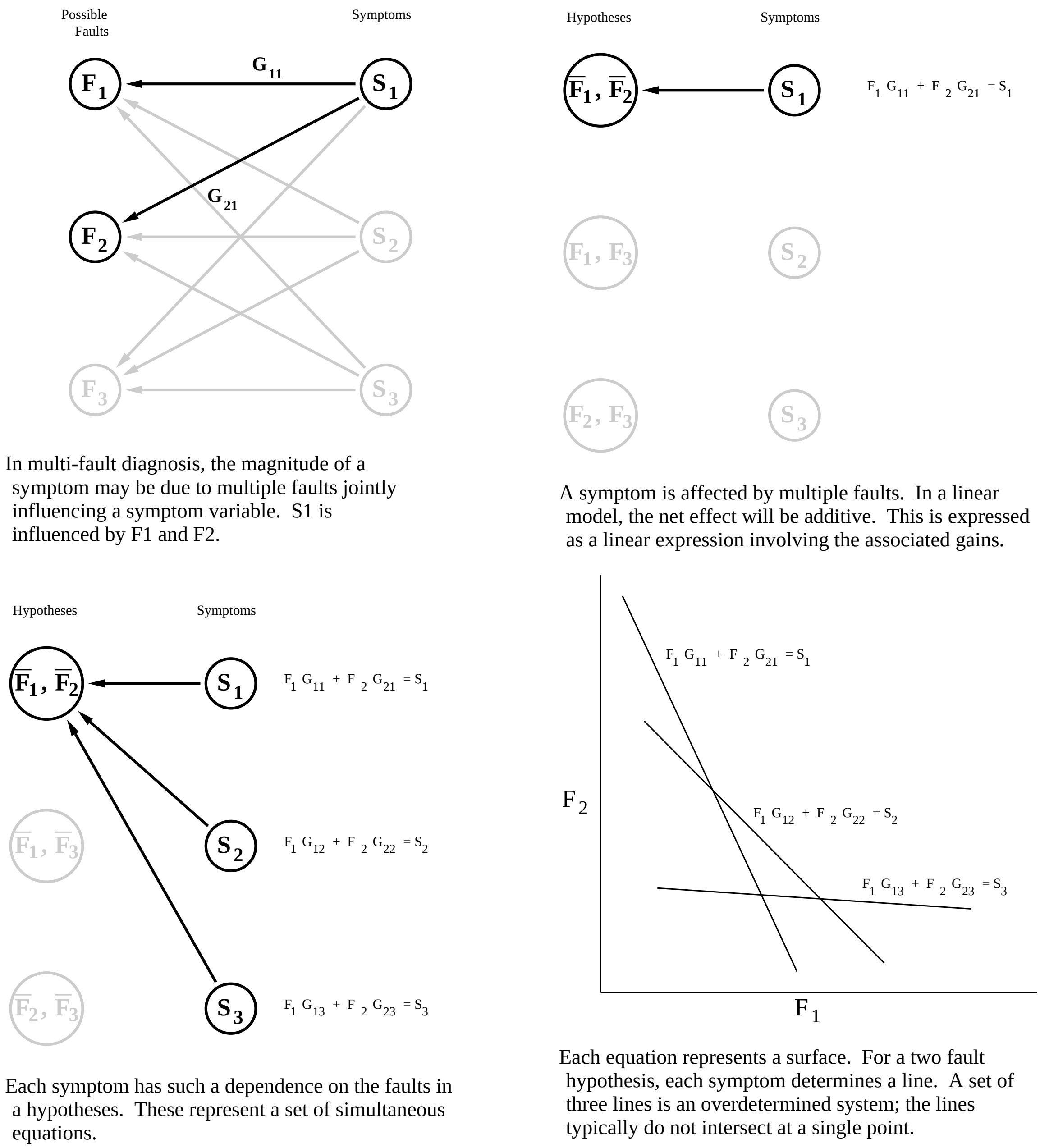


Figure 3.11: Pairs of symptoms estimate values for pairs of faults.

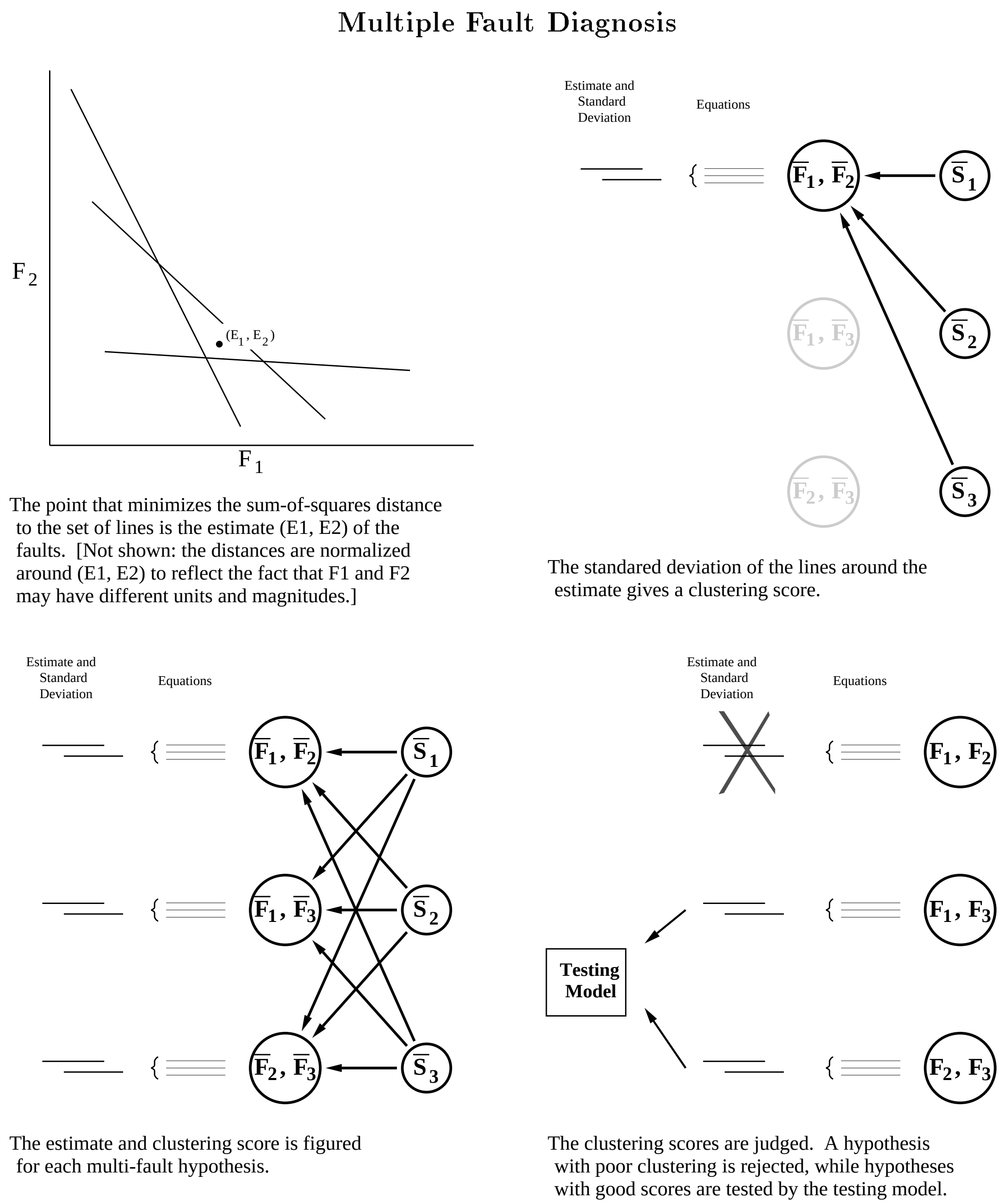


Figure 3.12: The estimates and standard deviations are calculated for all candidate hypotheses.

We will extend the clustering filter to two-fault hypotheses first, and then show the extension to N hypotheses. If $H_1 = F_1, F_2$, the clustering filter will provide deviation estimates $\overline{E}_1, \overline{E}_2$ and a standard deviation around the point $(\overline{E}_1, \overline{E}_2)$.

If there are three symptoms S_1, S_2 and S_3, there will be three equations

$$\begin{aligned} \overline{F}_1 G_{11} + \overline{F}_2 G_{12} &\approx \overline{S}_1 \\ \overline{F}_1 G_{12} + \overline{F}_2 G_{22} &\approx \overline{S}_2 \\ \overline{F}_1 G_{13} + \overline{F}_2 G_{32} &\approx \overline{S}_3 \end{aligned} \tag{3.1}$$

[Notice that, unlike the single fault case, some of the gains may be zero; $G_{ij} = 0$ when F_i is not causally upstream of S_j. This did not arise in the single fault case, as a fault needed to be connected to *all* symptoms to be in the candidate fault set.]

These three equations specify three lines in the F_1, F_2 plane, the *symptom lines.* If H_k were the correct hypothesis, and if the linear model exactly described the mechanism, these three lines would pass through a single point $(\overline{F}_1, \overline{F}_2) = (\overline{E}_1, \overline{E}_2)$. As the mechanism is not linear, the lines do not intersect, and the set of equations is overdetermined. This is shown graphically in Figure 3.11.

If the mechanism is adequately (but not exactly) modeled by the linear model, then (for the correct H_k) the lines will pass near $(\overline{F}_1, \overline{F}_2)$ but not through it. We seek an estimate $(\overline{E}_1, \overline{E}_2)$ that minimize the sum-of-squares distance to the symptom lines. The first panel in Figure 3.12 shows $(\overline{E}_1, \overline{E}_2)$ and its distance to three symptom lines.

The squared distance from a point (x_1, y_1) to a line $Ax + By + C = 0$ is

$$D^2 = \frac{(Ax_1 + By_1 + C)^2}{A^2 + B^2}. \tag{3.2}$$

We can put equations 3.1 into $A\overline{F}_1 + B\overline{F}_2 + C = 0$ form. There is a difficulty, though. The result will be a dimensional equation; A, B and C will have units. The estimates E_1 and E_2 will depend on our choice of units. This means that, for instance, if F_1 is a temperature and F_2 is a massflow, then if E_1 and E_2 are solved in °F and Mlbh, they will take a different value than if solved in °C and kg/sec.

We address this by normalizing equations 3.1, dividing each term by a characteristic value. The appropriate characteristic value for each fault is the estimate of the fault.[5]

[5]The other characteristic value, the nominal value of the fault, is inappropriate. Many unit scales

For each two-fault hypothesis H_k, with faults F_{k1}, F_{k2}, each symptom line

$$\overline{F}_{k1} G_{k1,j} + \overline{F}_{k2} G_{k2,j} = \overline{S}_j$$

is now rewritten as

$$\left(\frac{\overline{E}_{k1}\ G_{k1,j}}{\overline{S}_j}\right) \frac{\overline{F}_{k1}}{\overline{E}_{k1}} + \left(\frac{\overline{E}_{k2}\ G_{k2,j}}{\overline{S}_j}\right) \frac{\overline{F}_{k2}}{\overline{E}_{k2}} - 1 = 0. \tag{3.3}$$

This is a dimensionless form which fits into the distance formula, Equation 3.2, with

$$A = \frac{\overline{E}_{k1}\ G_{k1,j}}{\overline{S}_j} \qquad B = \frac{\overline{E}_{k2}\ G_{k2,j}}{\overline{S}_j} \qquad C = -1.$$

The distance squared to the j^{th} symptom line is then

$$D_j^2 = [\frac{\left(\frac{\overline{E}_{k1}\ G_{k1,j}}{\overline{S}_j}\right) \frac{\overline{F}_{k1}}{\overline{E}_{k1}} + \left(\frac{\overline{E}_{k2}\ G_{k2,j}}{\overline{S}_j}\right) \frac{\overline{F}_{k2}}{\overline{E}_{k2}} - 1]^2}{\left(\frac{\overline{E}_{k1}\ G_{k1,j}}{\overline{S}_j}\right)^2 + \left(\frac{\overline{E}_{k2}\ G_{k2,j}}{\overline{S}_j}\right)^2},$$

or, simplifying

$$D_j^2 = \frac{[\left(\frac{G_{k1,j}}{\overline{S}_j}\right) \overline{F}_{k1} + \left(\frac{G_{k2,j}}{\overline{S}_j}\right) \overline{F}_{k2} - 1]^2}{\left(\frac{\overline{E}_{k1}\ G_{k1,j}}{\overline{S}_j}\right)^2 + \left(\frac{\overline{E}_{k2}\ G_{k2,j}}{\overline{S}_j}\right)^2}.$$

For a fixed $\overline{E}_{k1}, \overline{E}_{k2}$ this can be expanded into the form

$$D_j^2 = b_{j1}\ \overline{F}_{k1}{}^2 + b_{j2}\ \overline{F}_{k1}\ \overline{F}_{k2} + b_{j3}\ \overline{F}_{k1} + b_{j4}\ \overline{F}_{k2}{}^2 + b_{j5}\ \overline{F}_{k2} + b_{j6} \tag{3.4}$$

where each b_j is a constant. For n symptoms, the standard deviation of the j symptom lines around the point $(\overline{E}_{k1}, \overline{E}_{k2})$ will be the sum of these squared distances, divided by n, so

$$n \cdot C_k = \sum_{j=0}^{n} b_{j1}\ \overline{F}_{k1}^2 + b_{j2}\ \overline{F}_{k1}\overline{F}_{k2} + b_{j3}\ \overline{F}_{k1} + b_{j4}\ \overline{F}_{k2}^2 + b_{j5}\ \overline{F}_{k2} + b_{j6} \tag{3.5}$$

$$= c_{k1}\ \overline{F}_{k1}^2 + c_{k2}\ \overline{F}_{k1}\overline{F}_{k2} + c_{k3}\ \overline{F}_{k1} + c_{k4}\ \overline{F}_{k2}^2 + c_{k5}\ \overline{F}_{k2} + c_{k6}$$

where each c_k is a constant. The minimum of this expression is found by setting the partials to zero

$$0 = \partial\overline{F}_{k1} = 2c_{k1}\ \overline{F}_{k1} + c_{k2}\ \overline{F}_{k2} + c_{k3}$$

have arbitrary zeros – if the fault is, say, a temperature, it would involve division by °F, which is not meaningful. Other scales that have arbitrary zeros include dates, elevation above sea-level, degrees longitude and gauge pressure. Additionally, although it does not occur in our example problems, for many fault parameters (such as leakage from a pipe) the nominal value is zero.

$$0 = \partial \overline{F}_{k2} = c_{k2}\ \overline{F}_{k1} + 2c_{k4}\ \overline{F}_{k2} + c_{k5}\ \overline{F}_{k2}$$

Solving for $\overline{F}_{k1}$ and $\overline{F}_{k2}$, we get two new estimates $\overline{E}_{k1-new}, \overline{E}_{k2-new}$ of $\overline{F}_{k1}, \overline{F}_{k2}$ – the values that minimize the least-squared distances to the set of symptom lines:

$$\overline{E}_{k1-new} = \frac{2c_{k3}c_{k4} - c_{k2}c_{k5}}{{c_{k2}}^2 - 4c_{k1}c_{k4}} \tag{3.6}$$

$$\overline{E}_{k2-new} = \frac{2c_{k1}c_{k5} - c_{k2}c_{k3}}{{c_{k2}}^2 - 4c_{k1}c_{k4}} \tag{3.7}$$

We set $(\overline{E}_{k1}, \overline{E}_{k2})$ to $(\overline{E}_{k1-new}, \overline{E}_{k2-new})$ and iterate at equation 3.3 until the values converge. Once they have converged, we have both a normalized estimate of the fault values in H_k and, by equation 3.5, a clustering score C_k by which to rank H_k's plausibility. The use of the least-squares joint estimate of the faults is shown graphically in Figure 3.12.

The foregoing illustrates the extension of the clustering filter to two fault hypotheses. The result can be generalized to any positive integer number of faults. If a linear function of N variables V is put into the form:

$$A_1V_1 + A_2V_2 + \ldots A_NV_N + A_{N+1} = 0$$

so that it specifies an $N-1$ dimensional surface in the N dimensions, then the distance is from a point outside the surface to the surface is given by

$$D^2 = \frac{A_1V_1 + A_2V_2 + \ldots A_NV_N + A_{N+1}}{A_1^2 + A_2^2 + \ldots A_N^2} \tag{3.8}$$

Equation 3.4 will be an equation quadratic in N variables with $2N$ terms. Taking its partials will again result in N linear equations in N unknowns. The solution to these equations will give $(E_1, E_2, \ldots E_N)$, the estimate of the hypothesis. The expansion of Equation 3.5 to N terms will give the standard deviation.

For the clustering filter to discriminate between hypotheses, there must be more symptoms than there are faults in the hypotheses. This assures that the set of equations generated from the symptoms and the gains (such as the ones in Equation 3.1) will be overdetermined. This implies that, generally, there will be no exact solution. If the system of equations is not overdetermined, for any set of fault variables there will be (in the linear approximation) some exact solution, the standard deviation will be zero, and the clustering filter will not be able to discriminate good hypotheses from bad ones.

Section 3.12 stated that HEATX contains no generator for variable identification for multiple faults. For this reason, the multiple-fault version of the clustering filter has not been integrated into a complete diagnostic program. However, it has been implemented as a running program. Its performance on selected multi-fault diagnosis problems will be discussed in Section 4.4.

3.13 Testing with Numerical Models

The generating model has identified a fault variable and made an estimate of its value. This information is sufficient to specify the inputs to the testing model.

With the estimate as input, we run the testing model. (Once again, we stress that we are assuming the fidelity of the testing model to the real world. Given an accurate and complete description of the fault, the model must generate a behavior that matches the observed faulty behavior.) There are three possibilities for the behavior that it generates:

- The fault hypothesis may be correct, and have been instantiated at a correct or nearly correct value. In this case, the model matches all the observed variables to within an acceptable range, and the hypothesis is accepted.
- The fault hypothesis may be correct, but the instantiated value may have significantly overshot or undershot the correct value. The system can iterate, looking for a closer value.
- The wrong fault hypothesis may have been picked. Either of the following is then possible:

  The program can find a value for this (incorrect) variable that produces the observed behavior. The incorrect hypothesis is completely consistent with the behavior and the diagnostic system can accept the wrong hypothesis.[6]

  Upon modeling the incorrect hypotheses (or after iterating on it several times) the diagnostic reaches a state in which it can make no improvement, but in which the modeled behavior does not match the observed behavior. The hypothesis is rejected.

If the variable identification on the hypothesis has been correct, and if the estimate has been close enough that the testing model can converge, the problem now becomes a standard variable tuning problem for the testing model.

[6]This is usually only a problem when the fault is mild and only one observable variable is driven out of tolerance.

There may be other, incorrect hypotheses which passed the filters. On these, the testing model may still succeed in producing a behavior that matches the observed fault behavior. In this cases, the true and false hypotheses are indistinguishable, at least on the basis of the observations we have here. We may be able to evaluate the hypotheses on other criteria – for instance, the U_o of a heat-exchanger rarely increases. We may also be able to analyze the two hypotheses to find competing predictions, either as a variable which can be observed with further effort or as the result of a perturbation experiment.

Normally though, when the variable identification is wrong the testing model will *disconfirm* it; the model will fail to find a variable value producing a behavior which matches the observed faulty behavior. The hypothesis is rejected. Normally, the correct fault hypothesis also survived the qualitative and clustering filters and will be the only hypothesis confirmed by the testing model.

It sometimes happens that the correct hypothesis fails the qualitative or clustering filters. The hypotheses that fail the filters were not discarded; they were saved in rank order for just such an event. We can return to these filters and select the hypothesis with the best score from among the failed hypotheses. We then run them until we find the correct hypothesis.

This will happen when the linear models of the mechanism are grossly off, and this in turn happens when the actual functions that describe the components' behaviors are grossly nonlinear. See Section 4.3 for an example of such a problem and how it is resolved.

It would be possible to accept a misdiagnosis. The true hypothesis could fail one of the filters and the testing model could falsely confirm some other hypothesis. We have not observed this to happen in any of the problems we have run.

A possibility of such a misdiagnosis is a shortcoming of our system. It is a potential problem for any generate-and-test diagnostic scheme in which not all possible hypotheses are tested and in which an incorrect hypothesis can sometimes pass the test.

In this report, the testing model is a numerical simulator using a component connection representation that iteratively calculates the steady state behavior of a heat-exchanger network.

We intend the approach described in this section to be applicable to any of a large number of possible numerical testing models, including dynamic simulators, stochastic

H_k	the k^{th} hypothesis
$\mathcal{F}$	the set of all component faults
$\mathcal{F}'$	the set of candidate faults
F_i	the i^{th} individual fault
F_{kN}	the N^{th} fault in H_k
S_j	the j^{th} individual symptom
G_{ij}	the gain between F_i and S_j
$\mathcal{V}$	the set of component variables
$\mathcal{V}_p$	the set of unvarying independent parameters
V_b	the b^{th} individual component variable
Γ_{ab}	the gain between V_a and V_b
T_j	the symptom subgraph for S_j
P_{ij}	the prediction of F_i from S_j and G_{ij}
$\mathbf{P}_i$	the vector of the signs of $\overline{P}_{i1}$ through $\overline{P}_{in}$
E_i	the estimate of F_i; the mean of P_{i1} through P_{in}
$C(F_i)$	the clustering score; the standard deviation of P_{i1} through P_{in} around E_i

Hatted variables, as in $\hat{F}_i$, indicate nominal values.
Overlined variables, as in $\overline{F_i}$, indicate deviation values.
Bracketed variables, as in $[F_i]$, indicate signs, $(+, -$ or $0)$

Table 3.3: Table of notation used in chapter 3.

simulations or analog models. We expect this work to be most applicable to numeric models built up using component connection representations. In Section 3.3 we listed the advantages of reasoning with single models. These advantages can be retained with separate generation and testing models if there is a one-to-one correspondence between the components in both models.

In a component connection representation for modeling, the model is normally built from a set of user-supplied specifications. The specifications list the components, their variables and their connectivity.

For our diagnostic system, we can build both the generating model and the testing model from the same specifications. There will be some information in the specification that is only used by one of the models, but the advantages we listed for reasoning with a single model – ease, consistency and task integration – remain.

Chapter 4

Implementation and Results in HEATX

Chapter 3 presented theoretical propositions about model-based diagnosis. These propositions have been implemented in a running computer program, HEATX. This program diagnoses failures in networks of heat-exchangers.

For the reasons that Section 3.12 explained, HEATX has been written to diagnose single-fault failures but the clustering filter (for estimating fault variables and evaluating hypothesis plausibility) has been extended to multiple-fault diagnosis. Sections 4.1 through 4.3 present the implementation and results of the single-fault diagnosis. Section 4.4 presents the implementation and results of the extension to multiple fault diagnosis.

4.1 Our Approach to Model-based Reasoning

Our approach is to build an automated diagnostic system that uses the following high-level algorithm:

1. Obtain a component-connection model for numerical simulation of the mechanism in question.
2. Enter the nominal values for the mechanism's independent variables into the model. Run the simulation to generate the mechanism's nominal behavior.
3. Obtain a set of observations of the mechanism. Compare the values of observable variables to their nominal values (from the simulation.) Identify observable variables that have departed from their nominal values. Call these the symptoms.
4. Use dependency-directed backchaining to identify candidate fault variables. Create hypotheses corresponding to these candidate faults.

5. Build a linear model to approximate the numerical simulation model in the neighborhood of the nominal operating point.

6. Filter the hypotheses on qualitative criteria. Reject as implausible those hypotheses that fail the qualitative tests.

7. For each remaining hypothesis, generate a set of predictions (one for each symptom) of the magnitude of the candidate fault variable.

8. Complete each hypothesis so that it provides an estimate of its fault variable. The estimate is the mean of the hypothesis' set of predictions.

9. Score the plausibility of each hypothesis. Take the score from the standard deviation of the predictions around the mean.

10. Select the best scoring hypotheses for instantiation into the numerical simulator.

11. Simulate each hypothesis, generating a numerical description of the behavior.

12. Match the behaviors against the set of symptoms. For each hypothesis and its generated behavior:

    (a) If the behavior matches well enough, accept the hypothesis. (Several hypotheses may be accepted, meaning that there are multiple explanations of the symptoms which are not distinguishable by the currently available observations.)

    (b) If the behavior does not match well, attempt to refine the hypothesis. Change the estimate and continue from step 11.

    (c) If the behavior does not match and refining the estimate cannot improve the match, reject the hypothesis.

13. If all hypotheses that were selected at step 10 (the clustering filter) have been rejected, lower the clustering criteria and continue from step 10.

14. If all hypotheses that were selected at step 6 (the qualitative filter) have been rejected, lower the qualitative criteria and continue from step 6.

We have implemented this algorithm in a computer program for reasoning about the domain of networks of heat-exchangers: HEATX. It uses:

- *Connectivity data*, which show how the mechanism's components (the individual pieces of equipment) interconnect.

- *Component knowledge*, including
  - Linear approximations of the component's outputs as a function of its inputs. This is expressed in the form of rules.
  - Component fault variables – a list of the variables in the component which could be faulty, (some variables of the model, such as the liquids' physical property data, are assumed not to possibly vary.)
- A set of *nominal values* for the input variables.
- A *numerical simulation subsystem*, which HEATX uses both
  - to predict the mechanism's nominal behavior, and
  - to confirm that a candidate fault is capable of causing the observed symptoms.

  The simulation routines in this subsystem are nonlinear, non-invertible, and (relative to the other routines in HEATX) expensive to run.
- A set of *field data* – a set of observations of the mechanism from its instrumentation (such as massflows, temperatures.)
- *Tolerances* for observations, which reflect inherent inaccuracies in measurement.

From these it builds:

- A *linear model* of the mechanism – an approximation to the more detailed numerical simulation. This linear approximation is constructed from the linear models of the components and connectivity data. This model is comprised of a set of fault/symptom *gains*: proportionality factors relating changes in candidate fault variables to changes in the observable variables.

4.1.1 Example: The Six Exchanger Network *(i)*

HEATX has been run on simulations of sixteen different mechanisms. The mechanisms evaluated were primarily networks of heat-exchangers, but also included networks of stirred heated tanks, stirred tank reactors and sensors and controllers. Of these, we have studied fault behaviors in one particular heat-exchange network in great detail. This network was used because its complexity is similar to the complexity of real

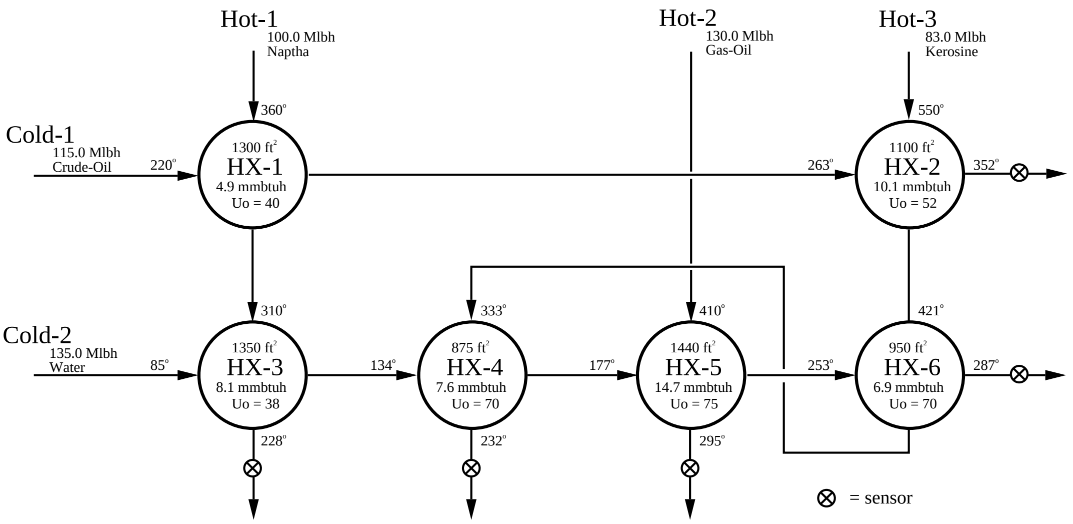


Figure 4.1: The Six-Exchanger Network

networks, and because its feedback loops and reconvergent fanout provide a good test of HEATX's ability to diagnose complex systems.

We will use this mechanism, a network of six heat-exchangers, as an ongoing example in this chapter as we explain the details of HEATX's implementation. Section 4.3 presents the results of diagnosing 32 different faults in this mechanism. Section 4.4 presents the results of diagnosing 16 double-fault failures in this same network. A schematic of the network is shown in Figure 4.1. The equations governing the heat-exchangers were given in Section 1.3.2.

This network has two cold streams and three hot ones. Nominal values for the inflow temperatures and massflow rates are given at the *battery limits*: the place where the streams enter the diagram. These nominal inflow values are known but not measured (meaning that one class of error that the system will consider is that these supplied values are wrong.) A surface area and nominal heat transfer coefficient (U_o) for each exchanger are known from design data. Temperatures for all streams are measured at their outflows (at the battery limits.)[1]

There are three types of fault this network can suffer. Any stream's inflow temperature, any stream's massflow rate, or any exchanger's U_o can depart from its nominal

[1]All temperature data is in degrees Fahrenheit. All massflow rates are in thousands of lbs/hr (Mlbh). All surface area is in ft^2. All heat transfer coefficients are in btu/hr/ft^2/°F. Physical property data for hydrocarbons from [Green, 1984].

value. And any departure can be either high or low. This gives us

$$2 \text{ directions} \times (6\ U_o + 5 \text{ inlet temperatures} + 5 \text{ massflows}) = 32 \text{ possible faults}$$

Because the only measurements taken are the five outflow temperatures, symptoms are the set of outflow temperatures that have departed from their nominal values by more than the tolerance, ±3.6°F.

4.1.1.1 Introducing the faults

In our example network, the fault which occurs is that the massflow rate for Cold-2 rises from 135 to 173 Mlbh. This will decrease the final temperatures of Hot-1, Hot-2, Hot-3 and Cold-2 itself; however, the final temperature of Hot-1 is not changed enough to register as a symptom. The symptoms noted, then, will be decreased final temperatures in the streams Hot-2, Hot-3 and Cold-2.

	Nominal Value	*Actual Value*
Fault Variable		
Cold-2 Massflow	135 Mlbh	173 Mlbh
Symptom Variables		
Cold-2 Outlet Temperature	287°	259°
Hot-2 Outlet Temperature	295°	282°
Hot-3 Outlet Temperature	232°	217°

We will look at this example fault repeatedly in this chapter.

4.2 Implementation

4.2.1 Specifics on the Encoding of HEATX

HEATX is runs on Symbolics computers under the Genera 7.2 operating system. It is implemented entirely in LISP. The program uses the Symbolics Flavors system for object-oriented programming. Otherwise, to the extent possible, it has been written in Common LISP. It is composed of four component subsystems, Units, Htex, Hypothesis and Mft. Table 4.1 shows the programs and their sizes.

Additionally, HEATX uses two graphics programs to display its output: POS [Throop, 1990] and POS-SKETCH. All figures in this report were created with these programs.

Component Subsystem	Description	Kilobytes of Source Files
Units	A program for doing calculations using numbers with units (e.g., 120°F, 25 btu/hr/ft^2.)	115
Htex	A program for simulating networks of heat-exchangers.	636
Htex:Networks;	Specifications of particular networks.	439
Hypothesis	A program for forming and testing hypotheses about the networks in Htex.	265
Mft	Routines extending the clustering filter to multiple fault diagnosis.	101

Table 4.1: Component subsystems of HEATX.

4.2.1.1 The Zero-Fault Simulation

Before any diagnosis can be done, HEATX must develop an expectation of the mechanism's nominal behavior. It simulates the mechanism, using the nominal values for the input variables of the mechanism. This gives the zero-fault simulation and the zero-fault behavior.

HEATX has a simulation subsystem for simulating chemical processing systems. This subsystem has a suite of programs which include a library of specifications and routines for simulating chemical processing equipment and related objects. Simulations in HEATX use the specifications for several object-classes:

- Heat-exchangers,
- Streams,
- Stream-sources and stream-destinations, (ports for streams to enter and leave the mechanism at the battery limits),
- Pipes, and
- Sensors.

The mechanisms which HEATX diagnoses are heat-exchange networks. The components of these networks are instances of these object-classes. The library includes methods (similar to subroutines) for these objects which specify how certain quantities are calculated (such as computing the outlet temperature for the hot stream of a heat-exchanger, or the pressure drop in a pipe.) The connectivity information for the network is encoded in the individual components; for instance, each heat-exchanger stores the names of the pipes which connect to it. There is also an object-class 'piping-system' which holds lists of all of the components in the network.

We were not able to obtain a physical network of heat-exchangers on which to practice diagnosis for this report. Therefore, HEATX keeps *two* numerical simulations of the network; the second serves as a proxy for a physical network. The field-data, instead of being obtained from physical sensors, are received from this simulation. The field-data, taken from component variables marked as 'observable,' are the only data taken from the proxy simulation. [In the discussion which follows, references to 'the simulation' refer to the first simulation, and not to the proxy, unless specifically noted otherwise.]

The simulator generates the steady state values for the mechanism's component variables. It does this through an iterative process. For each heat-exchanger in the network, it calculates the outlet temperatures as a function of the inlet temperatures, massflows and the heat-exchanger's independent variables (U_o, area, etc.) Changes in these outlets affect the inlets to other exchangers, which are recalculated in turn. In practice, four iterations over the network are usually enough to converge all temperature to within 1° of their final values, as long as the starting temperatures for the iteration were not too badly off. (If there is any doubt, the components have the capability of collecting the iterative values of their slots. These collected values can be shown graphically. These graphs give a good visual indication of how well the network has converged.)

Having run the zero-fault simulation, and having obtained the field data from the proxy simulation, HEATX compares the values for each of the observable variables in the mechanism. Associated with each type of sensor is a *tolerance*. If the sensed discrepancy between variable values in the zero-fault simulation and the field data exceed the tolerance, the discrepancy is counted as a symptom. For the results reported in this chapter, the tolerance was uniformly set at 3.6°F.

HEATX goes through the set of observable variables and generates a list of symptoms. It then begins the process of generating possible explanations for the symptoms.

4.2.2 Identifying Candidate Faults

Hypothesis formation in model-based diagnosis is symptom-driven. HEATX generates hypotheses by starting with the symptoms.

Section 3.6 gave the algorithm for generating candidate faults from the connectivity data and Section 3.8 discussed how the fault/symptom gains are built up from the component gains. In the implementation, these two processes are combined. A depth-first all-paths search is begun at each symptom, working backwards in the influence graph and building up a tree structure which is the symptom subgraph. A list of component slots seen in the search is kept on a separate global variable. Each new component slot seen is checked against the list. When the search encounters a component slot a second time, search is halted along that branch. Search also terminates whenever a candidate fault is encountered, or when independent variables are encountered.

As the symptom subgraph is built, the local component gains are computed and stored along the branches. This allows the symptom subgraph data structure to be used in two ways. It is used in conjunction with the subgraphs from the other symptoms in order to determine the set of candidate faults; these are the fault variables that appear as leaves in *all* the symptom subgraphs. The symptom subgraph data structure will be reused in building the fault/symptom gains, below.

The rules in HEATX express two kinds of information: the connectivity and the gain. In building up the symptom subgraph, HEATX uses only the connectivity information (although, as was said above, the component gains are stored in the symptom subgraph for later use.)

After generating all of the hypotheses that are causally linked to the symptoms, HEATX processes them further before simulating any of them. It discards most of the hypotheses that are initially generated on qualitative and approximate criteria.

4.2.3 Creating Hypotheses

Section 3.2 observed how hypotheses with continuous fault variables may be incomplete; they may include a qualitative specification on the fault variable ("higher than nominal") without providing a numeric estimate of the fault variable. A further refinement of this observation is that an incomplete hypothesis may have a range: the fault variable may be restricted to be between two values.

When a fault variable is in the symptom subgraphs of all symptoms, it becomes a candidate fault. HEATX generates two fault hypotheses for each of the candidate faults: one high, and one low. Initially, these hypotheses are incomplete. Associated with each is a range of possible values that the fault variable might take. For the low hypothesis, this ranges from the lowest value the variable could take up to the nominal value. For the high hypothesis, it ranges from the nominal value up to the highest value the variable could take, or up to infinity. In order to select hypotheses for simulation, these hypotheses are filtered. After the first, qualitative filter, the hypothesis will be completed; an estimate will be made of its fault variable's value, and it then will be sent to the clustering filter.

4.2.4 Linear Component Knowledge

Section 3.8 mentioned three ways in which component gains in the linear models of components could be obtained:

1. Using some fixed value for the component gains, and ignoring any changes that result from changing operating conditions, or
2. Calculating or estimating all the component gains each time the operating conditions change and storing them, or
3. Calculating them or estimating them when they are needed.

HEATX was originally implemented using (1), component gains were set by equipment class. For instance, the gain between the hot-inlet-temperature and the cold-outlet-temperature was set as $0.4\frac{°\mathrm{F}}{°\mathrm{F}}$ for all heat-exchangers, without making local calculations comparing the ratios of the specific heats or massflows, or the area of the exchanger, etc. Routines implementing approach (2) were added when the clustering filter was extended to multiple faults. No great improvement was noted, but because (2) seems inherently better based than (1), the approach (2) was retained and is the basis for the results in this chapter. It is noteworthy, though, that even the rough approach of (1) supplies enough information to discriminate between plausible and implausible hypotheses.

These component gains are translated into rules in the fashion described in section 3.8.

4.2.5 Building Fault / Symptom Gains from Component Models

The possible causes of the symptoms are determined by tracing back through the components, looking for candidate faults that lie upstream of all the symptoms. As this tracing proceeds, HEATX carries along the magnitude of the fault, inverting it through each component so that when it reaches a candidate fault, it will have (approximately) the magnitude of that fault needed to explain the observed symptom. Because the network is cyclic, the tracing must note when it reaches any place in the mechanism by more than one path, and combine these paths and their estimates.

Section 3.8 gave the algorithm for generating the fault/symptom gains from the individual component gains. The implementation in HEATX is faithful to that algorithm, with one additional detail. Many of the component variables have only one influence on them; these lead to long unbranching chains of rules in the influence graph. To translate each component variable in such a chain into its own equation would lead to a large, sparse matrix when solving the linear model. To avoid this, the set of equations in a chain are collapsed into a single gain from the top of the chain to the bottom, yielding a single equation, before being passed on to be solved.

The method does not build a linear model of the entire mechanism. It builds a linear model of the portion of the mechanism which lies along causal paths from candidate fault variables to symptoms. Component variables in the mechanism which do not lie along such paths either

- do not lie causally downstream of any candidate fault; this implies that they remain at their nominal values, and do not affect the symptoms, or
- do not lie causally upstream of any of the symptoms, and are not necessary to the modeling.

These gains are estimates of the partial derivative of the component's outputs with respect to its inputs. The program uses these gains in both its qualitative and its clustering filtering.

Several paths may branch out of a single node in the symptom subgraph. This represents a component variable which is influenced by a fault via multiple paths. This can be caused by a feedback loop, nondirectional components, or by reconvergent fanout; see the discussion in sections 2.3.4.2 and 3.12. These multiple influences are assumed to be simply additive.

The gains built up by this process are rough; there are several sources of error. The assumption of linearity is only a gross approximation to the actual case; if an incoming hot stream to a heat-exchanger both increases in its flow rate and gets hotter, the effect on the cold stream will be significantly more than the sum of each change alone. Even though the values for the component gains are rough, they still carry enough information that they usually yield the correct signs for the gains between the faults and the symptoms. HEATX breaks candidate generation into four stages, as was shown in Figure 3.3.

4.2.6 The Qualitative Filter

Section 3.6.1 discussed how to use the signs of the fault/symptom gains to qualitatively filter the hypotheses. For each fault variable, two hypotheses are generated, one 'low' and one 'high'. Each symptom supports a prediction for each candidate fault variable; symptoms for which that prediction is above the nominal value support the 'high' hypothesis (and contradict the 'low' one) and vice versa.

The discussion in Section 3.6.1 said that hypotheses which did not receive unanimous qualitative support were filtered. In HEATX, the count of the symptoms which qualitatively support a hypothesis is recorded on the hypothesis, and the filtered hypotheses are kept in ranked sets, with all hypothesis in a set having the same score. If all of the hypotheses having unanimous qualitative support are rejected, the next best ranking set will be passed to the qualitative filter.

We have noted that, by using the approximate information in the linear component gains, HEATX could resolve ambiguous confluences. However, in the presence of reconvergent fanout, this procedure can potentially defeat the qualitative filter. Suppose that some fault F_i affects two intermediate variables V_1 and V_2, with component gains of Γ_{i1} and Γ_{i2}. Suppose also that V_1 and V_2 both effect a symptom variable S_J, with component gains Γ_{1j} and Γ_{2j}. The fault/symptom gains between F_i and S_j is then

$$G_{ij} = \Gamma_{i1} \cdot \Gamma_{1j} + \Gamma_{j2} \cdot \Gamma_{2j}$$

If the two fault/symptom paths are opposite in sign but approximately equal in magnitude:

$$\Gamma_{j1} \cdot \Gamma_{1i} \approx -1 \cdot \Gamma_{j2} \cdot \Gamma_{2i}$$

then errors introduced by the linearization may predict an incorrect sign of G_{ij}. Indeed, the sign of G_{ij} may vary with the magnitude of F_i, or the nominal values of the operating conditions. In this case the qualitative filter may err. Although such errors do not arise in the examples we have considered, this is a part of the motivation for retaining the qualitatively filtered hypotheses for possible re-examination later in the diagnosis.

4.2.7 The Clustering Filter

After the qualitative filtering, there may be several hypotheses left that are in qualitative agreement with all of the observations. HEATX now uses the information in the fault/symptom gains to select the most plausible hypotheses.

Section 3.10 discussed how the symptom/fault gains are calculated; Figure 4.2 shows a graphical interpretation of a gain. The gain is shown as the slope of a line between a fault-axis and a symptom axis. The slope of this line, G_{ij}, projects a value sf the symptom S_j into a predicted value P_i of a fault.

In general, different symptoms may have different units, such as °F or Mlbh, (although all symptom variables in the system we are considering are temperatures.) We divide the magnitude of each symptom by its tolerance to allow all symptoms to be plotted on the same dimensionless axis. The top panel of Figure 4.3 shows how several symptoms predict a cluster of values for a fault variable. The median value of this cluster is the estimate E_j, the difference between the nominal value and E is $\overline{E}$. The sum of the squared distances gives the clustering score; for the single fault case, Equation 3.5 is just

$$C_j = \sum_{n=1}^{j}(\frac{P_n - E}{\overline{E}})$$

As figures 3.6 and 3.7 showed, the fault/symptom gains used by the clustering filter are derived from set of simultaneous equations which represent the component gains. These equations are solved in matrix form; all symptoms and gains are entered into the matrices (converted to their based units of lb-mass, ft, sec, °F) and the predictions generated (in the same base units.)

We did not monitor the condition number of the matrices. Because of the geometry of the mechanisms we considered, and because we were not considering stochastic variables, we did not judge such a check to be necessary, (but see the discussion of redundant symptoms in Section 4.4.2.) An extension of this method to noisy data would necessitate such a check in order to address the problem of nearly singular matrices.

The tendency to cluster is itself a continuous phenomenon, not a boolean one.

Each hypothesis is scored on how tightly its predictions cluster. The hypotheses are ranked by score, and some of them are selected as having a score good enough to justify further investigation. In choosing which hypotheses to investigate, we make a cutoff in this ranking. In our implementation, this has meant accepting the best scoring hypothesis, any hypothesis with a score not ten times worse than the best scorer, plus any hypothesis with a score better than a certain constant cutoff value.

4.2.7.1 The Six-Exchanger Example *(ii)*

Continuing the example introduced in Section 4.1.1, we follow the calculations for the symptoms in the six-exchanger network.

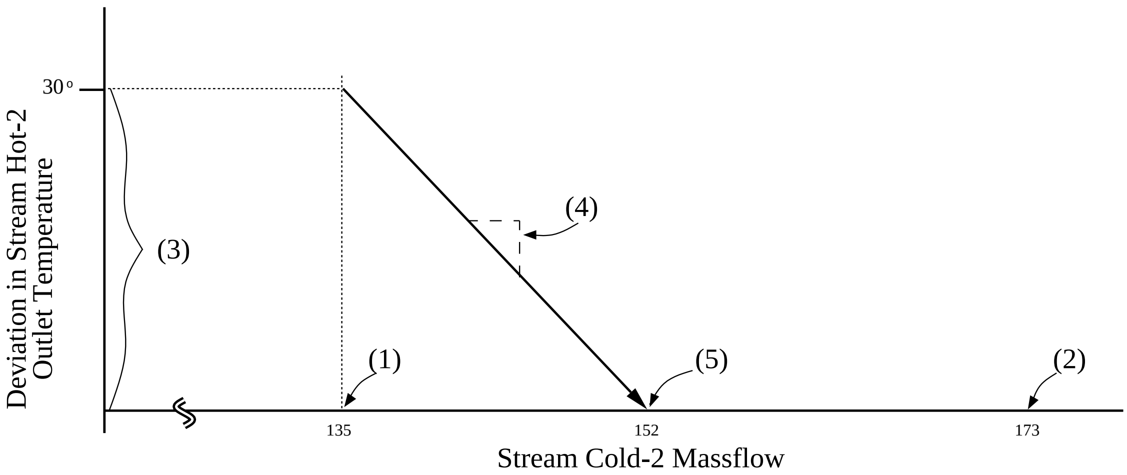
In the example mechanism the fault is that the massflow of Cold-2 rises from 135 to 173 Mlbh. One symptom of this is that the outlet temperature for Hot-2 falls 30°F below its nominal value.

The gain between the fault and the symptom is $-0.57\frac{°\mathrm{F}}{\mathrm{Mlbh}}$. In figure 4.2, the symptom and the gain predict that using a corrected value for the fault of 152 Mlbh for the massflow of Cold-2 will explain the symptom.

Figure 4.3 shows how each symptom predicts a value for the massflow of Cold-2. The average value of these predictions is the program's best guess for the actual value for the massflow of Cold-2.

In the six-exchanger network, because the massflow of Cold-2 is high, the outlet temperatures for streams Cold-2, Hot-2 and Hot-3 falls. The three symptoms make three predictions about the massflow of Cold-2. Their mean value of 153 Mlbh becomes the estimate for the hypothesis, as shown in Figure 4.3.

There are other faults which could cause qualitatively the same symptoms; for instance, if the inflow temperature of Cold-2 fell it would have qualitatively the same effect on the other streams. However, a plot of the predicted values shows a poorer clustering. Figure 4.4 shows this. HEATX concludes from this that the most plausible hypothesis is the one in which the massflow of Cold-2 is high, with its estimates of 153 Mlbh. Additionally one other hypothesis, that the massflow of Hot-3 has fallen from 83° to 65°, has a good enough clustering score that it is also selected for simulation.



The estimate of a fault is derived from an observation:

(1) The massflow for stream Cold-2 has a nominal value of 135 Mlbh.

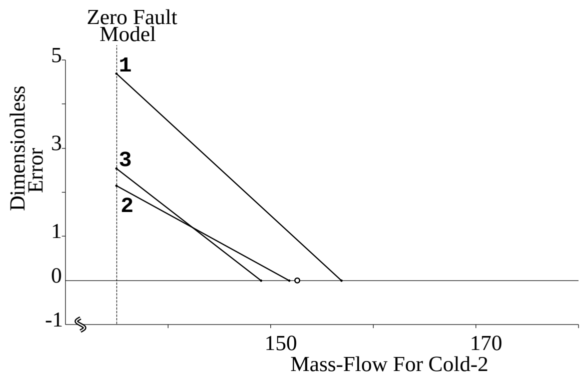
(2) The fault F_i is that the massflow for Cold-2 has risen from 135 to 173 Mlbh.

(3) One of the resulting symptoms S_j is that the outlet temperature for stream Hot-2 is 30° colder than its nominal value.

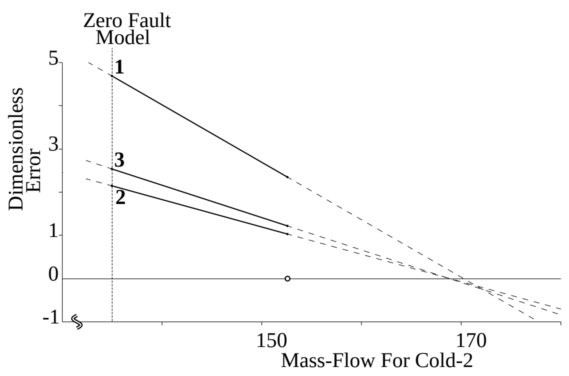
(4) The gain G_{ij} between the massflow for Cold-2 and the outlet temperature for Hot-2 has already been estimated to be -0.57$\frac{°\mathrm{F}}{\mathrm{Mlbh}}$. This gain is now applied to the the 30° error in the outlet temperature of Hot-2.

(5) The system makes a prediction P_i that the 30° error is caused by the massflow for Cold-2 being $135 + (-0.57 \times -30) = 152$ Mlbh.

Figure 4.2: Each Observation Predicts a Value for Faults

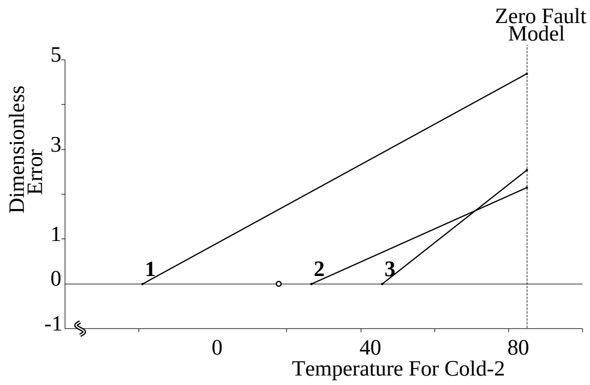


The (qualitatively correct) hypothesis shown here is that the massflow for Cold-2 is 153 Mlbh. This estimation is based on the gains from three symptoms: the outlet temperatures for the streams (1) Cold-2 (2) Hot-2, and (3) Hot-3.



A numerical simulation of this hypothesis has run. The match to the field data improve, but there are still unexplained residuals. Using gains estimated from the two simulations (from the zero-fault simulation and from the simulation checking the first estimate), a second estimation is made for the value of the massflow of Cold-2. This estimate of 169 Mlbh is close enough to the actual value that the next numerical simulation matches the field data, and the hypothesis is accepted.

Figure 4.3: Tight Clustering of Predictions Supports a Hypothesis.



The hypothesis shown here is that the inflow temperature forCold-2 is 18° (compared to the nominal value of 85°.) This estimation is based on the gains from three symptoms: the outlet temperatures for the streams (1) Cold-2 (2) Hot-2, and (3) Hot-3. The low degree to which these predictions cluster together excludes this hypothesis for further consideration.

Figure 4.4: Poor Clustering Makes this Hypothesis Implausible

4.2.8 Estimating Extreme Faults

We have discussed the plausibility of a hypothesis in relation to how well it fits a qualitative model, or a linear model, or a numerical simulation model. A hypothesis may also be implausible or even nonsensical on other grounds. In the heat-exchanger domain, some extreme temperatures are implausible, though the transition from plausible to implausible is gradual. For massflows and heat-transfers, there is a sharp line; negative massflows and negative heat-transfer-coefficients are physically impossible. But zero values for massflows and nearly zero heat transfers are characteristic of certain faults; even though negative values are meaningless.

Using linear gains to estimate faults with extreme values can give rise to estimates with meaningless values: when diagnosing a system with a badly fouled heat-exchanger, the estimate (from the clustering filter) of the exchanger's U_o may be negative, when it is actually zero. Such estimates require careful handling.

It would be a mistake to reject such a hypothesis just because it predicted an impossible value; likely values may lie close by. Instead the estimate should be set to the most extreme possible value, and the standard deviation should be taken around this limit value. This will increase the clustering score. However, it assures that, if the hypothesis is instantiated and entered into the simulation model, the values will be meaningful.

4.2.9 Testing with Numerical Simulation

In order to test each hypothesis, HEATX instantiates the estimate of the fault in the simulation model. That is, the simulation is rerun with all of the same inputs at their nominal values *except* that the (suspected) fault variable is set to the estimated fault value from the hypothesis. The program examines all three behaviors: the observed data, the zero-fault simulation, and the simulation with the fault.

There are three possibilities for a hypothesis:

- The fault hypothesis may be right, and have been instantiated at a correct or nearly correct value. In this case, the simulation of the hypothesis decreases the error for all observed variables to within their normal tolerances, and the hypothesis is confirmed.
- The fault hypothesis may be correct, but the instantiated value may have significantly overshot or undershot the correct value. If so, the magnitude of the error will have decreased, but still won't have matched within normal range. The program takes the data points from the zero-fault and the instantiated simulations, and uses Newton's method to converge on the correct value.
- The wrong fault hypothesis may have been picked. Then, either
  - Upon simulating the incorrect hypothesis (or iterating on it several times) the symptoms give qualitatively different predictions, and the hypothesis can be rejected.
  - The program can find a value for this (incorrect) variable that produces the observed behavior. The incorrect hypothesis is completely consistent with the behavior and HEATX can accept the wrong hypothesis.

  The second outcome is is usually only a problem when the fault is mild and only one observable variable is driven out of tolerance; HEATX may find several hypotheses acceptable, but it rarely if ever loses the right one.

4.2.10 Splitting Hypotheses and Qualitative Disagreement

After a hypothesis has been instantiated and numerically simulated, the output of the simulation may not match the symptoms. It is still possible that the hypothesis was qualitatively correct, but that it was instantiated at a value too far from the actual fault-value to generate a simulation that matched the symptoms. In such circumstances either

- the hypothesis can be instantiated at a new value, and the numerical simulation rerun, or
- the hypothesis can be discarded and some other one instantiated and run.

HEATX uses qualitative criteria to decide between these options. These criteria are extensions of the ones that were used to qualitatively distinguish the original hypothesis set. We have mentioned that each hypothesis is assigned a range. After a point has been instantiated and numerically simulated in that range, it is possible to split the range in two. For instance, consider a hypothesis that held that a fault variable is higher than nominal. Its original range was from the nominal value up to infinity. It has been instantiated at an estimate, and that estimate has been simulated. When it is split, the first of the two new hypotheses will range from the nominal value (for the fault variable) up to the simulated value; the second will range from the simulated value up to infinity (or the maximum possible value.) HEATX now looks at the qualitative estimates for each of the symptoms and qualitatively scores each new hypothesis on each of the symptoms – does each symptom predict that the fault variable lies in the hypothesis' range?

When the (parent) hypothesis was correct in its fault variable identification, all symptoms will support the same (new) hypothesis; the other will have no support and be discarded. Under such circumstances, this iteration is equivalent to Newton's method. When the (parent) hypothesis erred in its fault variable identification, the symptoms may support different new hypothesis. This is the same situation as the one that allows the qualitative filter to work: some symptoms indicate that the fault variables should be increased, some indicate it should be decreased. Both of the new hypotheses are rejected on qualitative criteria, and the simulator gives up on trying to improve the match to this fault variable.

We shall illustrate this idea with an example and a diagram.

4.2.11 The Six-Exchanger Example *(iii)*

(This continues an example that was introduced in sections 4.1.1 and 4.2.7.1.)

For instance, in the six-exchanger network example, the (correct) hypothesis was instantiated at 153 Mlbh and simulated. The second panel in Figure 4.3 shows that, although this value lessened the discrepancy for all three symptoms, it did not account for them. Having run two simulations, we now have two data points for each symptom: one for the fault variable at its nominal value of 135 Mlbh and one at the guess of 153 Mlbh.

We again generate an estimate for the fault variable; this time using the gains from the two data points rather than from those built up from the component gains. This procedure now predicts a massflow for Cold-2 of 169 Mlbh, as shown in the second panel of Figure 4.3. This is instantiated and run; it is close enough to the actual fault value of 173 Mlbh that the simulation generates a behavior that matches the field data. They hypothesis is accepted as an adequate explanation of the fault.

4.2.11.1 Discarding Explanations which Fail to Match

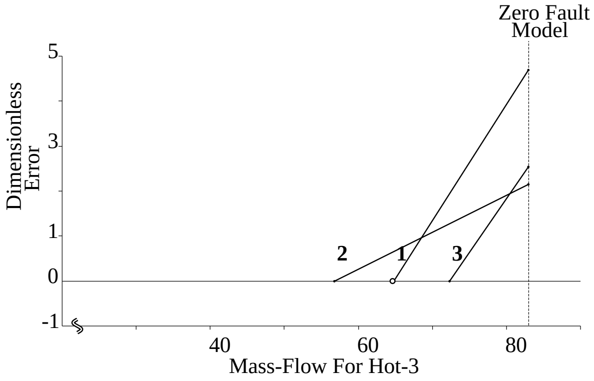
– In contrast, consider how the procedure handles the (incorrect) hypothesis that the symptoms are caused by the massflow of Hot-3 decreasing from 83 Mlbh to 65 Mlbh. Qualitatively this accounts for the symptoms; and the clustering filter does not exclude it, (although its score is worse than that of the correct hypothesis.) It is also instantiated and simulated. However, as Figure 4.5 shows, after one simulation it loses qualitative support and is not pursued further.

4.3 Results

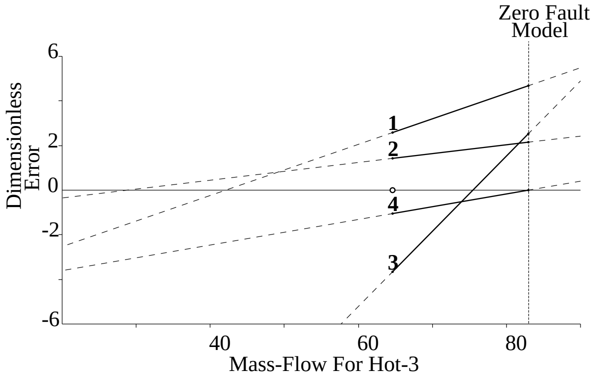
HEATX is able to solve several difficult real-world problems. For the six exchanger network in Figure 4.1. There are 32 possible qualitative classes of faults. We applied HEATX to an instance of each of these classes. The results are shown in table 4.2.

This network is as complex as a moderately large heat-exchanger train in an oil refinery. In each case, out of the 32 possible faults, HEATX first finds the faults which covers the symptoms, then filters the faults with the qualitative and clustering filters, then runs simulations of the remaining hypotheses. In all cases, HEATX finds the correct hypotheses.

The calls to the numerical simulator (for use as the testing model) are the most computationally expensive procedure in HEATX. The clustering filter improves computation by quickly filtering hypotheses for which simulation testing would be expensive. The performance listed in Table 4.2 does not indicate how much work the



This (incorrect) hypothesis – that the massflow of Hot-3 is 65 Mlbh – has reasonably good support from the clustering of its symptom's predictions.



However, when it is instantiated and simulated numerically, two things occur that contradict it. First, the simulation of the lower massflow for Hot-3 generates a symptom that was not observed in the field data: the outlet temperature of Cold-1 has decreased. Also, the four symptoms now point in two different directions for further improvements; with symptoms 1 and 2 indicating that the massflow of Hot-3 is below 65 Mlbh, and symptoms 3 and 4 indicating that it lies between 65 and the nominal value of 85 Mlbh. The hypothesis is split into two new hypotheses, with each one receiving the support of only two of the symptoms. Since neither of these has qualitative support as good as the remaining hypotheses (which include the correct hypothesis), neither is pursued further.

Figure 4.5: A Reasonable Hypothesis is Pursued and Disconfirmed.

	Equipment	Parameter	Nominal Value	Fault Value	Faults Found	Initial Symptom Count	Size Of Equiv. Class	Correct Hyp In First Equiv Class	Number Of Hypotheses Tested	Found Correct Fault	Correct Fault Is Unique
1	HX-1	U-Overall	40	26	Yes	2	1	Yes	1	Yes	Yes
2	HX-1	U-Overall	40	54	Yes	2	1	Yes	1	Yes	Yes
3	HX-2	U-Overall	52	38	Yes	2	1	Yes	1	Yes	Yes
4	HX-2	U-Overall	52	66	Yes	1	8	Yes	8	Yes	No
5	HX-3	U-Overall	38	24	Yes	2	1	Yes	1	Yes	Yes
6	HX-3	U-Overall	38	52	Yes	2	1	Yes	1	Yes	Yes
7	HX-4	U-Overall	70	56	Yes	1	16	Yes	16	Yes	No
8	HX-4	U-Overall	70	84	Yes	1	16	Yes	16	Yes	No
9	HX-5	U-Overall	75	61	Yes	2	1	Yes	1	Yes	Yes
10	HX-5	U-Overall	75	89	Yes	2	1	Yes	1	Yes	Yes
11	HX-6	U-Overall	70	56	Yes	1	16	Yes	17	Yes	No
12	HX-6	U-Overall	70	84	Yes	1	16	Yes	19	Yes	No
13	Cold-1	Mass-Flow	115	83	Yes	1	8	Yes	14	Yes	Yes
14	Cold-1	Mass-Flow	115	147	Yes	1	8	Yes	12	Yes	Yes
15	Cold-2	Mass-Flow	135	97	Yes	3	1	Yes	2	Yes	Yes
16	Cold-2	Mass-Flow	135	173	Yes	3	1	Yes	1	Yes	Yes
17	Hot-1	Mass-Flow	100	72	Yes	1	8	Yes	15	Yes	Yes
18	Hot-1	Mass-Flow	100	128	Yes	1	8	Yes	13	Yes	Yes
19	Hot-2	Mass-Flow	130	94	Yes	2	1	No	22	Yes	Yes
20	Hot-2	Mass-Flow	130	166	Yes	1	16	Yes	22	Yes	No
21	Hot-3	Mass-Flow	83	60	Yes	3	5	Yes	6	Yes	Yes
22	Hot-3	Mass-Flow	83	106	Yes	2	2	Yes	3	Yes	Yes
23	Cold-1	Temperature	220	178	Yes	2	1	Yes	1	Yes	Yes
24	Cold-1	Temperature	220	262	Yes	2	1	Yes	1	Yes	Yes
25	Cold-2	Temperature	85	43	Yes	4	1	Yes	1	Yes	Yes
26	Cold-2	Temperature	85	127	Yes	4	1	Yes	1	Yes	Yes
27	Hot-1	Temperature	360	318	Yes	2	1	Yes	1	Yes	Yes
28	Hot-1	Temperature	360	402	Yes	2	1	Yes	1	Yes	Yes
29	Hot-2	Temperature	410	368	Yes	2	1	Yes	1	Yes	Yes
30	Hot-2	Temperature	410	452	Yes	2	1	Yes	1	Yes	Yes
31	Hot-3	Temperature	550	508	Yes	2	3	Yes	3	Yes	Yes
32	Hot-3	Temperature	550	592	Yes	2	3	Yes	3	Yes	Yes

Table 4.2: Results of running all 32 faults in the six exchanger network.

	Equipment	Parameter	Nominal Value	Fault Value	Initial Symptom Count	Number of Simulations Run With Clustering Filter	Number of Simulations Run Without Clustering Filter
1	Hx-1	U-Overall	40	26	1	1	9
2	Hx-1	U-Overall	40	54	1	1	12
3	Hx-2	U-Overall	52	38	1	1	19
4	Hx-2	U-Overall	52	66	1	8	8
5	Hx-3	U-Overall	38	24	1	1	20
6	Hx-3	U-Overall	38	52	1	1	14
7	Hx-4	U-Overall	70	56	1	16	16
8	Hx-4	U-Overall	70	84	1	16	16
9	Hx-5	U-Overall	75	61	2	1	3
10	Hx-5	U-Overall	75	89	2	1	3
11	Hx-6	U-Overall	70	56	1	17	29
12	Hx-6	U-Overall	70	84	1	19	22
13	Cold-1	Mass-Flow	32	83	1	14	11
14	Cold-1	Mass-Flow	32	147	1	12	18
15	Cold-2	Mass-Flow	38	97	3	2	15
16	Cold-2	Mass-Flow	38	173	3	1	18
17	Hot-1	Mass-Flow	28	72	1	15	19
18	Hot-1	Mass-Flow	28	128	1	13	18
19	Hot-2	Mass-Flow	36	94	2	22	15
20	Hot-2	Mass-Flow	36	166	1	22	18
21	Hot-3	Mass-Flow	23	60	3	6	19
22	Hot-3	Mass-Flow	23	106	2	3	18
23	Cold-1-Source	Temperature	220	178	1	1	19
24	Cold-1-Source	Temperature	220	262	1	1	29
25	Cold-2-Source	Temperature	85	43	3	1	12
26	Cold-2-Source	Temperature	85	127	3	1	16
27	Hot-1-Source	Temperature	360	318	1	1	17
28	Hot-1-Source	Temperature	360	402	1	1	21
29	Hot-2-Source	Temperature	410	368	2	1	11
30	Hot-2-Source	Temperature	410	452	2	1	13
31	Hot-3-Source	Temperature	550	508	1	3	22
32	Hot-3-Source	Temperature	550	592	1	3	22

*These cases are an artifact of using an artificially perfect estimator; see explanation in text.

Table 4.3: Comparison of Diagnosis with and without the Clustering Filter.

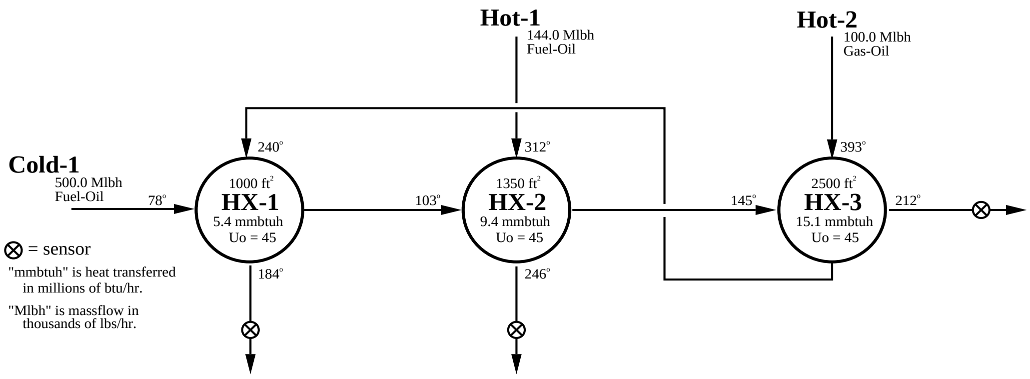


Figure 4.6: The Cold Outlet Temperature for Hx-1 Drops when the Flow in Cold-1 Becomes Laminar.

clustering filter is actually saving.

In order to assess the extent of this savings, the clustering filter was removed from HEATX and the same problem set as shown in Table 4.2 was rerun. Running without the clustering filter leads to a complication. The clustering filter not only filters implausible hypotheses; it also provides a fault estimate for plausible ones. When the clustering filter is removed, it is necessary to provide a fault estimate by some alternative means. If this alternative estimator is poor, it will exaggerate the savings from the clustering filter. If the alternative estimator is perfect, the comparison will still provide a lower bound on the savings provided by the clustering filter. Even if the alternative estimator is perfect, the clustering filter can still improve performance, as every hypothesis which is not filtered must still incur one call to the simulator.

HEATX was run with the clustering filter removed and with a routine which provided each hypothesis with an initial estimate of the fault magnitude. This 'estimator' provided the correct hypothesis with a perfect estimate (the estimator used the same code as used by the routine which created fault set.) This provided a lower bound on the work saved by the clustering filter. The results are shown in Table 4.3. Without the clustering filter, the number of calls to the simulator (over the 32 case fault set) rose from 207 calls to 608 call. Thus, the clustering filter saves approximately a factor of 3 on this test set.

For a few cases, the clustering filter did not remove any implausible faults. In some of these cases, HEATX with the filter removed outperformed HEATX with the filter in place; this is an artifact of using a perfect estimator of the fault in those cases.

4.3.1 Diagnosing Novel Faults

HEATX is also able to solve problems in which its heuristic information is incomplete or inaccurate. Normally, if the cold stream's flowrate into an exchanger falls, the cold stream's outlet temperature will rise; nearly the same amount of heat entering a smaller flow means a higher rise. However, if the flow decrease causes the cold stream's flow regime to change from turbulent to laminar, the heat transfer will degrade so greatly that the cold stream's outlet temperature will fall.

This is the situation in the mechanism shown in Figure 4.6. When HEATX is run with the general information about the relationships between flow and temperature, but with no knowledge of this exception, it initially attempts to prove other failure hypotheses for the mechanism. But it is unable to get the model to agree with these hypotheses, and eventually it attempts the "Cold Massflow Low" hypothesis, even though it disagrees qualitatively for some indicators. It instantiates the flowrate as low, runs the model, finds that it is making progress, runs the model again, and finally confirms the correct diagnosis.

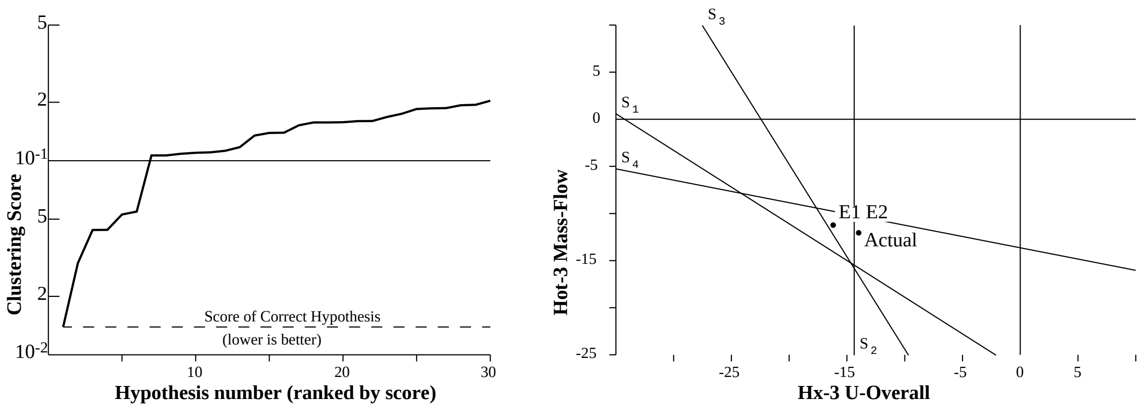
When HEATX finally simulates the correct hypothesis, the match to the observations is good. The detailed representation of the mechanism's components in the testing model allows the testing model to simulate this novel behavior. This is so even though the unusual response of the cold stream temperature to "Cold Massflow Low" was not represented in the generating model. We see then, that model-based diagnostic systems need not have explicit mention of all the abnormal situations and their manifestations in order to perform diagnosis.

4.4 The Extension of the Clustering Filter to Multiple Faults

(This section continues an ongoing example which began in Section 4.1.1. A diagram of the network appears in Figure 4.1.)

4.4.1 The Six-Exchanger Network *(iv)*

We apply this to the six-exchanger example. Suppose that two faults occur in the network:



Left panel: The clustering scores for the best 30 two-fault hypotheses. The score of the actual faults is shown by the dotted line. *Right panel:* The estimate of the correct fault hypothesis and its distance to the four symptoms, S_1 through S_4. The clustering score of the hypothesis is the sum of the squares of the distance of the estimate (E_1, E_2) from the symptom lines.

Figure 4.7: Clustering Score for Two-Fault Hypothesis

- U_o for HX-3 falls from 38 to 24 btu/hr/ft^2/°F, and
- the massflow for the stream Hot-3 drops from 83 to 71 Mlbh.

This produces four symptoms. The streams affected, and their observed outlet temperatures, are

Stream	*Nominal value*	*Observed value*	*Change*
Cold-2	278°	291°	13°
Hot-1	249°	225°	−24°
Hot-2	300°	308°	8°
Hot-3	250°	281°	31°

The fault/symptom gains are calculated in the same way as in the single fault case. For these faults and symptoms, they are

	Symptom	*Fault Variable* HX-3 U_o	Hot-3 Massflow
S_1	Cold-2 Temperature	$51.69e^{-3}$ °F^2 sec^3/lb	1.66 °F sec/lb
S_2	Hot-1 Temperature	$-239.52e^{-3}$ °F^2 sec^3/lb	0.00 °F sec/lb
S_3	Hot-2 Temperature	$50.33e^{-3}$ °F^2 sec^3/lb	0.642 °F sec/lb
S_4	Hot-3 Temperature	$78.69e^{-3}$ °F^2 sec^3/lb	8.26 °F sec/lb

The four symptoms produce four equations of the form

$$\overline{F}_{k1} \cdot G_{k1,j} + \overline{F}_{k3} \cdot G_{k2,j} = \overline{S}_j.$$

These lines are shown in the right panel of Figure 4.7. The estimate derived from these lines is shown in the figure as the point $\overline{E}_1, \overline{E}_2$, the point with the minimum distance to the four symptom lines. The actual fault value, $\overline{F}_1, \overline{F}_2$ is shown for comparison.

In the six-exchanger network, there are 16 different fault variables, so there are $\binom{16}{2} = 120$ possible fault pairs. Of these, 92 fault pairs are set-covers for the three symptoms. All of these fault pairs were run by the MFT routines in HEATX, and an estimate and a clustering score made for each pair. The left panel of Figure 4.7 shows the distribution of scores for the best 30 of these, ranked by score. As shown by the dotted line in that panel, the correct hypothesis scored lower, (that is, better) than any other score.

Two groups of two-fault problems were run under the multi-fault clustering filter. In the first group, consisting of ten problems, the magnitudes of the faults were kept relatively small, with massflow changes within ±25%, with temperature changes within ±50° and with U_o changes within ±50%. In this group, the rankings of the clustering filter were very good: the correct fault-pair was ranked higher than all others over half the time; and it was never ranked below third place.

In the second group, consisting of five problems, the relative sizes of the faults were increased. For each of these problems, the fault-pair includes at least one massflow that has decreased by more than 80%. In these problems, the discrimination of the multifault clustering filter decreased drastically, with rankings for the correct hypothesis as low as 40^{th} among 74 covering hypotheses.

4.4.2 Redundant Symptoms

Equations 3.6 and 3.7 gave the estimates E_1, E_2 with the lowest clustering score:

$$\overline{E}_{k1-new} = \frac{2c_{k3}c_{k4} - c_{k2}c_{k5}}{{c_{k2}}^2 - 4c_{k1}c_{k4}}$$

$$\overline{E}_{k2-new} = \frac{2c_{k1}c_{k5} - c_{k2}c_{k3}}{{c_{k2}}^2 - 4c_{k1}c_{k4}}$$

The equations involve a division. The denominator of these equations may go to zero; in such a case there is no estimate which minimizes the clustering score. This happens when the symptom lines are parallel. The physical interpretation of such values is that all symptoms are related to all of the faults through the same intermediate values. An instance of this occurs in the example of the six-exchanger network. The outlet temperatures of the streams Hot-2, Hot-3 and Cold-2 are all causally downstream of both the Cold-2 inlet temperature and the HX-3 U_o. All causal paths between these possible faults and these symptoms run through the Cold-2 flow between HX-3 and HX-4. Deviations in either fault variable have identical profiles in the symptom variables.

The effects of the two faults (in the Cold-2 inlet temperature and the HX-3 U_o), are indistinguishable. If neither of these faults (at any value) could singly account for the symptoms, then no combination of the two could, either. The values in equations 3.6 and 3.7 are tested during the scoring of multiple fault hypotheses. When a zero-valued denominator is detected, the hypothesis is rejected because of the redundancy among its faults.

4.5 Chapter Summary

This chapter has described the implementation of HEATX and provided results from problems run on HEATX. We have seen that

- The theory presented in chapter 2 has been implemented in a running program, HEATX.
- The details of this implementation were discussed, including program organization, knowledge representation, and performance optimizations.
- Results were presented for running instances of all 32 possible single-faults on a network of 5 streams and six exchangers. The system always succeeded in

finding the correct fault. HEATX performed best on faults which caused multiple symptoms.

- Results were presented for application of the methodology to a mechanism displaying a novel manifestation of a fault: an unexpected rise of cooling water outlet temperature when the cooling water massflow fell. Again, HEATX managed to diagnose the system.

- Results of extending the clustering filter to multiple-fault diagnosis were presented.

  Performance was reviewed on ten problems in which the magnitude of the faults were small to moderate. The filter worked well on these problems, ranking the correct two-fault hypothesis lowest (best) half the time, and never ranking the correct hypothesis worse than third.

  Performance was reviewed on five problems in which the magnitude of the fault was much larger. The performance of the filter fell badly on these problems.

Chapter 5

Discussion, Conclusions, Directions for Further Work and Summary

5.1 Discussion

There are many kinds of mechanisms in the world, and there are myriad methods of diagnosing them. With the advent of computers, people have attempted to encode many of these methods into computer programs. Many of the methods are numeric, and the computer's power at crunching numbers has made it an attractive tool for diagnosis. But the results of automating diagnosis have often been often disappointing. Early attempts led to single-use diagnostic systems: programs which performed well on some small, fixed set of problems, but which were difficult to adapt to similar problems, which didn't scale up to larger problems, and to which new knowledge could be added only with difficulty. These programs dealt well with the numerical complexity of the problems, but they were inflexible.

The field began to realize that the bottleneck in automated diagnosis was not numerical calculation. Rather, it was a lack of diagnostic theory with which to order decision making and the flow of control within diagnosis. Researchers went back to the basics of reasoning and wrote new kinds of programs which replaced numerical calculation with inferencing. These new programs, first the rule-based systems and then model-based ones, showed impressive performance and great flexibility. But having traded numerical computation for logical inferencing exacted a price. There are diagnostic problems which cannot be solved without subtle numerical computation. The inferencing procedures that these programs encoded, (and the diagnostic theory which underlay these procedures) had no place for complex numerical calculation routines.

This is not to say that no numerical computation was done; virtually every such program had a 'backdoor' by which the diagnostic routine could escape into a general programming language (usually LISP) in order to calculate some needed number. But such escapes were not part of the diagnostic theory.

The initial motivation for this dissertation was the desire to bring numerical computation back into the theory of diagnosis: to provide a basis for placing numerical information in the data structures and in the propagation and inferencing routines. It seemed that there were phenomena which had repeatedly caused difficulties for expert systems and for model-based reasoning, and that the roots of the difficulty lay in the weak representation for numerical information. Feedback systems and nonlinear systems were among those phenomena; the interactions in such systems are too complex to be adequately expressed in anything less than numerical representations.

Chapter 3 provides a theoretical basis for such programs. Chapter 4 describes the implementation of one particular program HEATX, and reports results from running it on a variety of problems. This section will review both HEATX's successes and its shortcomings. It will also suggest improvements which could improve HEATX's performance, and review some of the design decisions that were made in implementing HEATX. We shall also consider whether the performance we have reported justifies the claims for this report made in Section 1.1.4.

HEATX showed good performance in three areas, diagnosing

- 32 single-fault problems in a network of six heat exchangers and five streams (detailed in Section 4.3); and
- Ten two-fault problems in the same network, in which the deviation of the stream massflows was kept to less than 25% (detailed in Section 4.4.)

The six-exchanger network in these examples had both feedback loops and reconvergent fanout. Problems involving either of these phenomenon require solving several constraints simultaneously. Section 2.3.4.2 discussed how local propagation schemes fail to solve such problems; recall the comment there from Davis and Hamscher [Davis and Hamscher, 1988] which described how current constraint systems fail because of their 'local' propagation methods.

The several abstraction steps in HEATX that compose the fault/ symptom gains from the component gains (linearization of the components, the formation of the symptom subgraphs, the extraction of simultaneous equations from the symptom subgraphs, the solution of the equations to yield the fault/symptom gains) represent an alternative approach to the constraints which encode the components' behavior. This approach replaces the original constraint set with a set of linearized approximations, and then solves this (overdetermined) linearized constraint sets simultaneously.

We see no way in which the dozens of component connections (including feedback paths and reconvergent fanout) which typically lie between a fault and a symptom could be meaningfully aggregated *without* a numerical component to the representation. The fact that HEATX succeeded in diagnosing these example problems in the presence of these phenomena demonstrates the soundness of an explicitly numerical approach.

The performance on the 32 single-fault problems was somewhat uneven; the performance was much better on problems showing two or more symptoms. This unevenness supports the view that the arbitration of numerical evidence is a powerful technique in diagnosis: in single-symptom problems there is only one line of evidence and no arbitration occurs. The distribution of results (from Figure 4.2) was

Number of Symptoms	*Average Number of Simulations Until Solution*	*Number of Cases*
1	15.2	10
2	2.6	17
3	3.0	3
4	1.0	1

HEATX also solved

- One single-fault problem with a novel manifestation in a network of three exchangers and two streams (detailed in Section 4.3.1.)

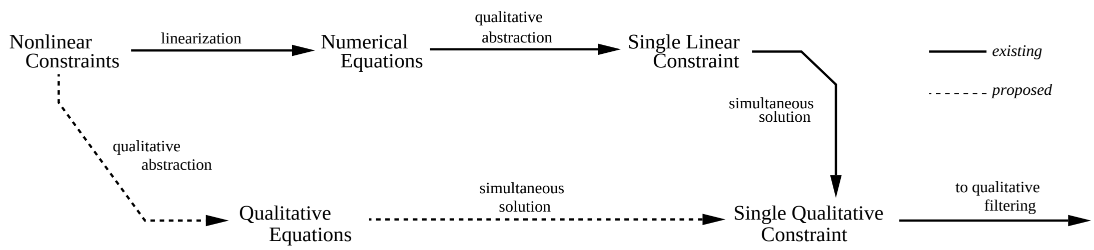
This was intended to be a much harder problem than any of the other 32 single-fault problems, and the program wandered around for a while, attempting other solutions, before attempting (and confirming) the correct hypothesis. The solution space of hypotheses in this problem small enough that even a random search might have eventually stumbled upon the right answer. The indication that HEATX found the correct hypothesis by design and not by luck is seen by examining the result carefully. Although the flow became laminar in the first and led to unexpectedly poor heat-transfer there, the flow did not become laminar in the other two heat-exchangers, and so their performance was consistent with the expected result of low massflow in the coldstream. Therefore, this hypothesis did have *some* support, and was investigated earlier than other possible hypotheses. And when the hypothesis was instantiated into the numerical simulation model, the simulator correctly simulated the (unexpected) drop in heat-transfer, thereby improving the match and leading to the correct diagnosis.

The component models are linearized as part of hypothesis generation. This resulting linearized mechanism model is a simplification of the original numerical model. The linearized model supports reasoning from symptoms to their possible causes (which the original model does not) but some of the knowledge encoded in the original model has been simplified out of the linearization. The change in heat-transfer under laminar flow conditions is an example of such knowledge. A known class of fault (low massflow) has an unexpected manifestation in this case; the manifestation is unexpected because the knowledge was not transferred into the simplified model. However, HEATX still manages to solve this single-fault problem. The fault had several manifestations (symptoms); not *all* of them were novel, and these other symptoms were handled correctly during hypothesis generation. HEATX then found some support for the correct hypothesis, and therefore passed the hypothesis to the simulation model. The knowledge about the relationship between laminar flow and heat transfer *was* encoded in the testing model, and so HEATX was able to confirm the correct hypothesis.

HEATX also showed shortcomings; diagnostic performance degraded badly on

- Five two-fault problems in the six-exchanger network, in which the deviation of the stream massflows was larger than 80% (detailed in Section 4.4.)

The methods for arbitrating evidence depend upon the methods for propagating values from the symptoms back through the components to the candidate faults. The propagation methods used by HEATX do not assume that the component behavior is linear. However, they do assume that the component behaviors are at least roughly approximated by linear models. It appears that in the case of problems with drastic changes in massflow, this assumption is violated. The approach of propagating numerical values may still work if some extension beyond linear component models is introduced. In particular, as the behavior at extreme deviations seems to be causing the most difficulty, a representation which captured the shape of the curve of the component behaviors seems promising. Using such higher-order derivative knowledge is discussed in Kuipers *et al* [Kuipers *et al.*, 1991]; an approach by Stephanopoulos [Stephanopoulos, 1990] of describing such functions in terms of bracketing triangles maintains numeric information while providing a description that is composable and invertible.



If the qualitative abstraction were performed earlier, the sign of the result of the qualitative abstraction would be ambiguous where it is now potentially incorrect.

Figure 5.1: Alternative Method for the Qualitative Filter

A potential problem with the qualitative filter was described but not seen in the problems we considered: reconvergent fanout can defeat the sign algebra inside the qualitative filter (as discussed in Section 4.2.6.) The error can occur when the equations (derived from the component linearizations) are solved simultaneously to give the fault/symptom gains: because the equations are only approximations, if several effects on a variable nearly cancel, the sign of the fault/symptom gain may be incorrect. In such cases, if the qualitative abstraction applied before the solution of the equations (instead of afterwards) the result of the solution of the equations would be of ambiguous sign, rather than of incorrect sign. Such a method could avoid this potential problem; it is shown schematically in Figure 5.1.

5.1.1 Further Extensions of the Clustering Filter

Section 3.12 gave the mathematical basis for the extension of the clustering filter to multiple-fault diagnosis. The technique was extended first to two-fault, then to N-fault diagnosis. In single-fault diagnosis, the predictions of fault values are points in the one-dimensional space of possible fault values F_i. In two-fault diagnosis, the predictions become lines in the two-dimensional plane of $F_{k1} \times F_{k2}$. In three-fault diagnosis, the predictions become planes in the 3-space of $F_{k1} \times F_{k2} \times F_{k3}$ and so on. The amount of computation involved in solving Equation 3.8

$$D^2 = \frac{A_1 V_1 + A_2 V_2 + \ldots A_N V_N + A_{N+1}}{A_1^2 + A_2^2 + \ldots A_N^2}$$

is linear in N the number of terms. Equation 3.5

$$n \cdot C_k = c_{k1}\ \overline{F}_{k1}^2 + c_{k2}\ \overline{F}_{k1}\overline{F}_{k2} + c_{k3}\ \overline{F}_{k1} + c_{k4}\ \overline{F}_{k2}^2 + c_{k5}\ \overline{F}_{k2} + c_{k6}$$

gave the formula for computing the location of the estimate with the minimum standard deviation for two-fault diagnosis; when extended to N-fault hypotheses it grows linearly with N and takes an amount of computation which is linear with N. However, as we go to higher numbers of faults, the simultaneous occurrence of these faults becomes less and less likely, unless the faults are correlated. If the faults *are* correlated, the likelihood is that the fault values will also be correlated, and should be estimated jointly, not independently. Diagnosis of correlated faults is beyond the scope of this report.

If there is no evidence that the faults are correlated, a reasonable procedure for diagnosis is to first attempt to explain the symptoms in terms of a single fault. Only if all single-fault hypotheses fail should multiple fault hypotheses be explored.

The *number* of N-fault hypotheses grows exponentially with N. Thus, even though the amount of computation in exploring *one* hypotheses grows linearly with N, the total effort of multi-fault diagnosis is still exponential.

5.1.2 Implementation decisions

HEATX represents all faults as variables which have deviated from their nominal values. Actually, *some* faults have magnitudes and are well-represented as independent variables which have deviated from their nominal values; some (such as mis-assembly) are not. The clustering analysis, which gives HEATX much of its diagnostic power, depends upon this representation.

Section 1.2 mentioned that choosing this representation is equivalent to having explicit fault models of components. This requires more knowledge and more coding effort: fault variables must be included in the constraints so that they describe the faulty mechanism as well as the nominal one. Choosing to use fault models has important consequences for HEATX's inferential power. Section 2.3.3.1 indicated that there different advantages in reasoning either with or without fault models of components. Among the advantages to reasoning without fault models was the improved ability to reason about faults of mis-assembly, or in which components acquire unexpected capabilities.

The decision to reason about faults which have numerical values led to a representation which used fault models for components. To give HEATX the ability to reason about mis-assembly, or Ungar's problem in diagnosing the presence of catalyst in a heat exchanger, would involve integrating HEATX's knowledge representation with a representation that does not use component fault models. This would be an interesting problem for further research.

HEATX uses the symptoms as evidence for hypothesis formation. Not all observed variables are symptoms, and the fact that some observed variables have *not* left their nominal values is evidence against some hypotheses. We notice that, in the testing phase, HEATX repeatedly simulates incorrect hypotheses which manage to match the behavior of the symptom variables well, but drive some non-symptom variable away from its nominal value.

It may be possible to extend the clustering filter to use the evidence in these unchanged variables and to thereby improve the discrimination of the filter. The extension will demand some care, however. In a fault in a larger system, when there are less than ten symptoms there may be tens of thousands of unchanged variables. Deciding which of these unchanged variables to include as evidence to the clustering filter will demand more than a trivial extension.

There are other representations which allow numerical information to be represented explicitly, and yet also provide support for model-based reasoning; the QSIM program, with its recent extensions to mixed quantitative/qualitative representations Q2 and Q3 [Kuipers and Berleant, 1988], is prominent among these. Although we are very familiar with the QSIM representation, we did not incorporate it into HEATX for several reasons:

- At the time we began encoding HEATX, the inclusion of a quantitative representation in QSIM was just beginning. QSIM at that time could not completely represent the numerical information which HEATX needed.
- We have been interested in the use of multiple knowledge representations and multiple models, and the challenges attendant upon translating between them. Q2 and Q3 have pursued the goal of capturing both qualitative and quantitative information in a single, unified representation, sidestepping the problem of moving amongst multiple representations.
- This dissertation has focused on the inclusion of fully numeric information into a methodology that has not greatly used numeric information in the past. QSIM's greatest strength lies in its representation of qualitative, non-numeric data. As that was not the focus of this report, it did not seem worthwhile to build the HEATX representation on top of the QSIM representation.

Nonetheless, we emphasize that our experience in working with QSIM has been extremely important in forming the ideas expressed in this report, especially in the understanding of the filtering of hypotheses.

We have presented a methodology which unites a numeric model with model-based reasoning. The work can also be described as a methodology in more general terms:

- Choose a component-based representation.
- Abstract and approximate components so that
    (a) the component models are invertible, and
    (b) they can be composed into invertible mechanism models.
- Compose invertible mechanism model. Generate fault candidates by any convenient method, (variations on dependency directed backtracking are a good default.)
- Regress symptoms through model to fault, arriving with quantitative information; use agreement to judge fault hypothesis.
- Pass to numeric model.

We will consider the application of this approach to other numerical representations in Section 5.3.

Automated diagnosis of systems with continuous variables have had to choose between the imprecision of AI methods and the inflexibility of non-AI methods; Section 2.2.1.2 mentioned that AI representations have always provided escapes into a general programming language. In this way, these methods have always had a numeric computation capability "through the backdoor." But this capability did not fit into any place in the theory of automated, intelligent diagnosis. Only a small fraction of the automated diagnosis done in industry incorporates the techniques that AI research has provided; most automated diagnosis is still done using FORTRAN programs which make no distinction between knowledge and inferencing. This will remain true until AI techniques provide explicitly techniques for handling the numerical information on which these methodologies turn.

The HEATX program represents a real break with the past AI practice of shoehorning numerical values into symbolic abstractions; instead it leaves the numeric information in the representation and reasons with it explicitly.

5.2 Conclusions

There has been scant work extending model-based diagnosis to incorporate fully numeric representations. Further, such work has mostly been limited to consideration of mechanisms which are adequately represented by linear models; in these cases the calculation has been straightforward. Without using fully numeric representations, model-based diagnosis has had difficulty modeling mechanisms with continuous variables, with feedback (and similarly, nondirectional components and reconvergent fanout), and with nonlinear components.

This report has described the extension of model-based diagnosis to a domain in which these features are present. It has reported successful diagnoses for single-fault failures and for a major subclass of multi-fault failures. It has done this by using a knowledge representation which supports quantitative reasoning about components and mechanisms. The incorporation of quantitative reasoning in model-based diagnosis was supported by several insights:

- Hypotheses should include estimates of the values of continuous variables.

The data structure which encodes the hypotheses should encode these estimates explicitly, and their values should be available to, and used by, the inferencing routines.

- Faults can be characterized by profiles: the gain between the fault and the symptom variables, and the ratios of magnitudes of multiple symptoms.

For nonlinear mechanisms, these profiles change with the magnitude of the fault and the operating conditions. Even so, the profiles change slowly enough that matching the observed symptom profiles against the expected profiles from a fault is a good check of the plausibility of a fault hypothesis.

- The clustering filter provides a quantitative judgment of how well the profile of a set of symptoms match the value expected from a fault hypothesis.

The score from the clustering filter provides a way of ranking the plausibility of hypotheses; this score is calculated in conjunction with calculating the best estimate of the fault value for a hypothesis.

- When there are multiple faults, the profiles which are characteristic of a single fault combine into more complex profiles. The profiles of individual faults can be extracted from these combined profiles if the faults' deviations from the nominal value are not too large.

The extension of the clustering filter to multi-fault problems provides a new approach to the problem of handling faults whose manifestations interact, canceling or obscuring each other. The filter's performance on multiple small faults is good, but falls sharply in the presence of drastic faults. As such it complements other methods which perform better on gross faults than small ones, (see [Mah, 1990, p. 423]).

- Profiles of the response of a mechanism can be composed from linear approximations of component response.

The choice of linear approximations assures that the component models will be invertible, and that they can be composed together to give an approximation of the mechanism's behavior.

- Single fault hypotheses can be filtered qualitatively, even before the estimated value for the fault has been calculated.

Qualitative filtering can reduce calculation; it can also prune spurious hypotheses which the clustering filter would not reject.

- Interleaving generation and testing reduces computation.

In HEATX, a hypothesis can be subjected to three tests: the qualitative filter, the clustering filter and the numerical simulation. Using fast, simple tests early allows us to prune hypotheses even before the generation process is complete.

5.3 Directions for further research

This dissertation has extended the knowledge representation used in model-based reasoning; the methodology of this extension has enabled the diagnosis of a class of steady-state mechanisms, using a steady-state numerical simulation system. Section

5.1.2 outlined a generalization of this methodology, intended to be applicable to other numerical models. There are many different numerical representations available; we anticipate that it would be fruitful to repeat this approach on a series of such models, including

Laplace transform models	Fourier transform models
Finite element simulations	Dynamic simulations
Continuous parameter models	Models incorporating Kalman filters with noise.

In HEATX a hypothesis is given a clustering score, which is a measure of the plausibility of the hypothesis. The plausibility of a hypothesis may also be judged probabilistically; with judgment based upon Bayesian inference, using knowledge of the prior probability of the hypothesis being true and other probabilistic descriptions of the state of the mechanism.

These are two rather different plausibility measures, but there are circumstances under which it would be useful to combine them. For instance, if the clustering score gave poor support to a hypothesis 'Hot Stream massflow 25 Mlbh Low,' but there was a high prior probability associated with low massflow in that stream (perhaps from a pump known to have suffered some damage) then it might be worthwhile to run the 'Hot Stream massflow 25 Mlbh Low' hypothesis through the numerical simulation anyway, based on the combined support from probabilistic reasoning and the clustering score. Combining support in such a way would involve putting the clustering score and the probabilistic measurements on some common basis. Such a basis would pave the way for combining reasoning from both associative inference and quantitative calculation.

HEATX uses linear approximations to model components because they are invertible and because they are composable. Variable values are measured as symptoms and traced back through the mechanism, inverting their values through the component models. A completely different approach is to represent the components by their algebraic constraints and to collect algebraic expressions as the causal paths are traced through the mechanism. HEATX uses a very simple algebra system to collect the coefficients for the simultaneous equations when generating the fault/symptom gains. A more advanced algebra system might permit better component descriptions than linear approximations, and thereby enhance the ability of HEATX to resolve multiple faults, even when well away from the linear region around the nominal value.

Over the years, in many different fields, a great deal of effort has gone into developing high-fidelity numerical simulation models. Potentially, these existing models can be used by model-based diagnosis systems, enabling diagnosis to be automated with much less effort (and much more accuracy) than if the models were to be built from scratch. To tap this potential, the methods of knowledge representation used in model-based diagnosis must be capable of expressing and reasoning with the same kinds of knowledge that these models encode: quantitative knowledge about mechanisms. This dissertation has provided a model-based framework for representing this knowledge and for performing inference upon it.

BIBLIOGRAPHY

[Abbot *et al.*, 1987] Kathy Abbot, Paul Schutte, Michael Palmer, and Wendell Ricks. Faultfinder: A diagnostic expert system with graceful degradation for onboard aircraft applications. In *Proceedings of the 14th Symposim 'Aircraft Integrated Monitoring Systems'*, September 1987. (Friedrichshafen, Germany).

[Abbot, 1988] Kathy H. Abbot. Robust operative diagnosis as problem solving in a hypothesis space. In *AAAI-88*, 1988.

[Anderberg, 1973] Michael R. Anderberg. *Cluster Analysis for Applications*. New York, Academic Press, 1973. Originally presented as the author's thesis, University of Texas at Austin, 1971.

[Bareiss and Farquhar, 1986] Ray Bareiss and Adam Farquhar. Fault diagnosis using qualitative information. Technical Report AI TR86-25, Artificial Intelligence Laboratory, The University of Texas at Austin, 1986.

[Brachman and Lesveque, 1985] Ronald J. Brachman and Hector J. Lesveque, editors. *Readings in Knowledge Representation*. Morgan Kaufman, 1985.

[Center, 1990] Boeing Computer Services Artificial Intelligence Center. Environmental control and life support system advanced automation project; 1990 annual report, November 1990. Contract no: NAS 8-37200.

[Chandrasekaran and Mittal, 1982] B. Chandrasekaran and Sanjay Mittal. Deep versus compiled knowledge approaches to diagnostic problem solving. In *AAAI-82*, 1982.

[Chiu, 1988] Charles Chiu. Higher order derivative constraints and a QSIM-based total simulation scheme. Technical Report AITR88-65, Artificial Intelligence Laboratory, The University of Texas at Austin, 1988.

[Cohen, 1985] Paul Cohen. *Heuristic Reasoning About Uncertainty*. Pitman, 1985.

[Collins, 1989] John W. Collins. Building qualitative models of thermodynamic processes. In *Qualitative Physics Workshop*, 1989.

[Dalle Molle, 1989a] D. T. Dalle Molle. Automatic evaluation of third order constraints in qualitative simulation. Technical Report AI89-98, Artificial Intelligence Laboratory, The University of Texas at Austin, February 1989.

[Dalle Molle, 1989b] D. T. Dalle Molle. *Qualitative Simulation of Dynamic Chemical Processes*. PhD thesis, Dept of Chemical Engineering, University of Texas at Austin, May 1989.

[D'Ambrosio, 1990] A. D'Ambrosio. Modeling real-world processes: Deep and shallow knowledge integrated with approximate reasoning in a diagnostic expert system. In Michael L. Mavrovouniotis, editor, *Artificial Intelligence in Process Engineering*, pages 189–222. Academic Press, 1990.

[Davis and Hamscher, 1988] Randall Davis and Walter Hamscher. Model-based reasoning: Troubleshooting. In Howard E. Shrobe, editor, *Exploring Artificial Intelligence: Survey talks from the National Converences on Artificial Intelligence*, pages 297–346. Morgan Kaufman, 1988.

[Davis *et al.*, 1984] Randall Davis, Bruce G. Buchanan, and Edward H. Shortliffe. Production ruses as a representation for a knowledge-based consultation program. In William J. Clancey and Edward H. Shortliffe, editors, *Reading in Medical Artificial Intelligence*, pages 98–130. Addison-Wesley, 1984.

[Davis, 1985] R. Davis. Diagnostic reasoning based on structure and behavior. In D. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. MIT Press, Cambridge, Mass, 1985.

[de Kleer and Brown, 1984] J. de Kleer and J.S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–84, December 1984.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1):97–130, 1987.

[de Kleer *et al.*, 1990] Johan de Kleer, Alan K. Mackworth, and Raymond Reiter. Characterizing diagnosis. In *AAAI-90*, 1990.

[de Kleer, 1977] Johan de Kleer. Multiple representations of knowledge in a mechanics problem-solver. In *IJCAI-77*, pages 299–304, 1977.

[de Kleer, 1985] Johan de Kleer. How circuits work. In D. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. MIT Press, Cambridge, Mass, 1985.

[de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, pages 127–162, 1986.

[DeCoste, 1990] Dennis DeCoste. Dynamic across-time measurement interpretation. In *AAAI-90*, page 373 ff, 1990.

[Doyle *et al.*, 1991] Richard J. Doyle, Usama M. Fayyad, Danile Berleant, Leonard K. Charest, Jr., Luiz S. Homem de Mello, Harry J. Porta, and Mark D. Wiesmeyer. Sensor selection in complex system monitoring using information quantification and causal reasoning. In B. Faltings and P. Struss, editors, *Recent Advances in Qualitative Physics*. MIT Press, 1991.

[Dvorak and Kuipers, 1989] Daniel Dvorak and Benjamin Kuipers. Model-based monitoring of dynamic systems. In *IJCAI-89*, pages 1238–1243, 1989.

[Dvorak, 1987] Daniel L. Dvorak. Expert systems for monitering and control. Technical Report AI87-55, Artificial Intelligence Laboratory, The University of Texas at Austin, May 1987.

[Fagan *et al.*, 1984] Lawrence M. Fagan, Edward H. Shortliffe, and Bruce G. Buchanan. Computer-based medical decision making: From MYCIN to VM. In William J. Clancey and Edward H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*, page 241 ff. Addison-Wesley, 1984.

[Finch, 1989] Francis Eric Finch. *Automated Fault Diagnosis of Chemical Process Plants Using Model-based Reasoning.* PhD thesis, MIT, Dept of Chemical Engineering, August 1989.

[Forbus, 1981] Ken Forbus. Qualitative reasoning about physical processes. In *IJCAI-81*, 1981.

[Forbus, 1984] K. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, December 1984.

[Fouche and Kuipers, 1990] Pierre Fouche and Benjamin Kuipers. Introducing energy into qualitative simulation. Technical Report AI90-139, Artificial Intelligence Laboratory, The University of Texas at Austin, September 1990.

[Gaschnig, 1979] John Gaschnig. Preliminary performance analysis of the prospector consultant system for mineral exporation. In *IJCAI-79*, 1979.

[Genesereth, 1985] Michael R. Genesereth. The use of design descriptions in automated diagnosis. In D. Bobrow, editor, *Qualitative Reasoning about Physical Systems*. MIT Press, Cambridge, Mass, 1985.

[Gentner and Stevens, 1983] Dedre Gentner and Albert Stevens, editors. *Mental Models*. Lawrence Erlbaum Associates, 1983.

[Glass *et al.*, 1990] Brian Glass, Bill Erickson, and K. J Swanson. TEXSYS: A large scale demonstration of model-based real-time control of a space station subsystem.

In *First International Workshop on Principles of Diagnosis*, pages 49–58. AAAI, 1990.

[Grantham and Ungar, 1990] Stepen D. Grantham and Lyle H. Ungar. A first principles approach to automated troubleshooting of chemical plants. *Computers and Chemical Engineering*, March 1990.

[Grantham and Ungar, 1991] Stepen D. Grantham and Lyle H. Ungar. Comparative analysis of qualitative models when the model changes. Ben gave this to me., 1991.

[Grantham *et al.*, 1990] Stepen D. Grantham, Catherine A. Catino, and Lyle H. Ungar. Automatic generation of qualitative models of chemical process units. *Computers and Chemical Engineering*, October 1990. Submitted.

[Green, 1984] Don W. Green, editor. *Perry's Chemical Engineers' Handbook*. McGraw-Hill, 6 edition, 1984.

[Griener *et al.*, 1989] R. Griener, B. A. Smith, and R. W. Wilkerson. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence*, 41:79–88, 1989.

[Hamscher, 1988] Walter C. Hamscher. *Model-Based Troubleshooting of Digital Systems*. PhD thesis, MIT, 1988.

[Hayes, 1979] P.J. Hayes. Naive physics manifesto. In D. Michie, editor, *Expert Systems in the Micro-Electronic Age*. Edinburgh University Press, 1979.

[Hayes, 1985] P.J. Hayes. Naive physics I: Ontology for liquids. In Hobbs and Moore, editors, *Formal Theories of the Commonsense World*, pages 71–106. Ablex, 1985.

[Himmelblau, 1978] David M. Himmelblau. *Fault Detection and Diagnosis in Chemical and Petrochemical Processes*. Elsevier Scientific, 1978.

[Hoskins and Himmelblau, 1988] J. C. Hoskins and D. M. Himmelblau. Artificial neural network models of knowledge representation in chemical engineering. *Computers and Chemical Engineering*, 12(9/10):881–890, 1988.

[IFA, 1978] *Identification and system parameter estimation: Proceedings of the 4th IFAC Symposium, Tbilisi, USSR*. Internation Federation on Automated Control, 1978. IFAC Symposium on Identification and System Parameter Estimation.

[Jamieson *et al.*, 1984] John R. Jamieson, Ethan Scarl, and Carl Delaune. NASA's LOX expert system. In *Proceedings of the AIAA Meeting, Computers in Aerospace 5, New Orleans, LA*, 1984.

[Kay, 1991] Herbert Kay. Monitoring and diagnosis of multi-tank flows using qualitative reasoning. Master's thesis, The University of Texas at Austin, May 1991.

[Klein and Finin, 1987] David Klein and Tim Finin. What's in a deep model? A characterization of knowledge depth in intelligent safety systems. In *AAAI-87*, pages 559–562, 1987.

[Kramer, 1990] Mark A. Kramer. Process systems diagnosis: Theory and practice. In *First International Workshop on Principles of Diagnosis*. AAAI, 1990.

[Kuipers and Berleant, 1988] B. Kuipers and D. Berleant. Using incomplete quantitative knowledge in qualitative reasoning. In *IJCAI-88*, 1988.

[Kuipers and Chiu, 1987] B. Kuipers and C. Chiu. Taming intractable branching in qualitative simulation. In *IJCAI-87*, 1987.

[Kuipers and Kassirer, 1984] Benjamin Kuipers and Jerome P. Kassirer. Causal reasoning in medicine: Analysis of a protocol. *Cognitive Science*, 8:363–385, 1984.

[Kuipers *et al.*, 1991] Benjamin Kuipers, Charles Chiu, David T. Dalle Molle, and David R. Throop. Curvature constraints in qualitative simulation. *Journal of Aritificial Intelligence*, 51:343–379, 1991.

[Kuipers, 1984] B. Kuipers. Commonsense reasoning about causality: Deriving behavior from structure. *Artificial Intelligence*, 24:169–204, December 1984.

[Kuipers, 1986] B. Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, December 1986.

[Leonard and Kramer, 1990] J. Leonard and M. A. Kramer. Improving the backpropagation algorithm for training neural nets. *Computers and Chemical Engineering*, 14(3):337–341, 1990.

[Mah, 1990] Richard Mah. *Chemical Process Structures and Information Flows*. Buttersworth, 1990.

[Malin *et al.*, 1987] Jane T. Malin, Bryan D. Basham, and Richard A. Harris. Config project: A generic tool for qualitative-simulation-based reasoning about configurations of engineered systems. In *Proceedings, Second Workshop on AI & Simulation, AAAI-87*, July 1987.

[Malin *et al.*, 1990] Jane T Malin, Bryan D. Basham, and Richard A. Harris. Use of qualitative models in discrete event simulation to analyze malfuntions in processing systems. In Michael L. Mavrovouniotis, editor, *Artificial Intelligence in Process Engineering*, pages 81–121. Academic Press, 1990.

[Mavrovouniotis and Stephanopoulos, 1989] Michael L Mavrovouniotis and George Stephanopoulos. Order of magnitude reasoning with o[m]. *Artificial Intelligence and Engineering*, 4(3):106–114, 1989.

[McDermott, 1982] D. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science*, 6:101–155, 1982.

[Melle,] Melle. *The EMYCIN Manual.*

[Newell and Simon, 1963] A. Newell and H. A. Simon. GPS, a program that simulates human thought. In Feigenbaum and Feldman, editors, *Computers and Human Thought*, pages 279–269. McGraw-Hill, 1963.

[Newell, 1973] Allen Newell. Aritificial intelligence and the concept of mind. In Roger C. Schank and Kenneth Mark Colby, editors, *Computer Models of Thought and Language*, pages 1–60. W.H. Freeman and Sons, 1973.

[Newell, 1981] Allen Newell. The knowledge level. *AI Magazine*, Summer 1981.

[Ng, 1990] Hwee-Tou Ng. Model-based, multiple fault diagnosis of time-varying, continuous physical devices. In *Proceedings of the Sixth IEEE Conference on Artificial Intelligence Applications*, pages 9–15, 1990.

[Norman, 1983] Donald A. Norman. Some observations on mental models. In Dedre Gentner and Albert Stevens, editors, *Mental Models*. Lawrence Erlbaum Associates, 1983.

[Oyeleye, 1990] Olayiwola Oluwemimo Oyeleye. *Qualitative Modeling of Continuous Chemical Processes and Applications to Fault Diagnosis*. PhD thesis, MIT, Dept of Chemical Engineering, February 1990.

[Park, 1979] Sunwon Park. *Fault Detection and Diagnosis in Chemical Process Plants*. PhD thesis, University of Texas at Austin, 1979.

[Patil *et al.*, 1984] R.S. Patil, P. Solovitz, and W.B. Schwartz. Causal understanding of patient illness in medical diagnosis. In W.J. Clancey and E.H. Shortliffe, editors, *Readings in Medical Artificial Intelligence*. Addison Wesley, 1984.

[Patil, 1981] Ramesh S. Patil. *Causal Representation of Patient Illness for Electrolyte and Acid Base Diagnosis*. Massachusets Institue of Technology, 1981.

[Patil, 1988] Ramesh S. Patil. Artificial intelligence techniques for diagnostic reasoning in medicine. In Howard E. Shrobe, editor, *Exploring Artificial Intelligence: Survey talks from the National Converences on Artificial Intelligence*, pages 347–379. Morgan Kaufman, 1988.

[Piela *et al.*, 1991] P. C. Piela, T. G. Epperly, K. M. Westerberg, and A. W. Westerberg. ASCEND: An object-oriented computer environment for modelling and analysis: The modeling language. *Computers and Chemical Engineering*, 15(1):53–72, 1991.

[Pople Jr., 1982] Harry E. Pople Jr. Heuristic methods for imposing structure on ill-structured problems: The structuring of medical diagnostics. In Peter Szolovits, editor, *Artificial Intelligence in Medicine*, pages 119–190. Westview Press, Boulder CO, for the AAAS, 1982.

[Raiman, 1986] Olivier Raiman. Order of magnitude reasoning. In *AAAI 86*, pages 100–104, 1986.

[Reiter, 1987] R. Reiter. A theory of diagnosis from first principles. *AI Journal*, 32(1), 1987.

[Roddis, 1988] W. M. Kim Roddis. *Heruistic, Qualitative and Quantitative Reasoning About Steel Bridge Fatigue and Fracture.* PhD thesis, MIT, Dept of Civil Engineering, 1988.

[Ross, 1990] Gavin J. S. Ross. *Nonlinear Estimation.* Springer-Verlag, 1990.

[Scarl *et al.*, 1985] Ethan A. Scarl, John R. Jamieson, and Carl I. Delaune. A fault detection and isolation method applied to liquid oxygen loading for the space shuttle. In *IJCAI-85*, pages 414–416, 1985.

[Scarl *et al.*, 1988] Ethan A. Scarl, John R. Jamieson, and Edwin New. Model-based reasoning for diagnosis and control. *Florida AI Research Symposium*, 1988.

[Schutte *et al.*, 1987] Paul Schutte, Kathy Abbot, Michael Palmer, and Wendell Ricks. An evalueation of a real-time fault diagnosis expert system for aircraft applications. In *Proceedings of the 26th IEEE Conference on Decision and Control*, December 1987.

[Shortliffe, 1976] T. Shortliffe. *Computer-Based Medical Consultations: MYCIN.* American Elsevier, 1976.

[Simmons and Davis, 1987] Reid Simmons and Randall Davis. Generate, test and debug: Combining associational rules and causal models. In *IJCAI-87.* Morgan Kaufman, 1987.

[Skorstad and Forbus, 1989] Gordon Skorstad and Ken Forbus. Qualitative and quantitative reasoning about thermodynamics. In *Qualitative Physics Workshop*, 1989.

[Sorenson, 1980] Harold W. Sorenson. *Parameter Estimation, Principles and Problems.* Marcel Dekker, Inc, 1980.

[Stephanopoulos, 1990] G. Stephanopoulos. Artificial intelligence in process engineering—current state and future trends. *Computers and Chemical Engineering*, 14(11):1259–1270, 1990.

[Struss and Dressler, 1989] Peter Struss and Oskar Dressler. "Physical Negation" – Integrating fault models into the general diagnositic engine. In *IJCAI-89*, pages 1318–1323. Morgan Kaufman, 1989.

[Throop, 1987] David R. Throop. Getting rules to talk to models. In J.S Kowalik, editor, *Proceedings of the Second Workshop on Coupling Symbolic and Numerical Computation.* North Holland, 1987.

[Throop, 1990] David R. Throop. Pos user's manual. Technical Report AI90-120, Artificial Intelligence Laboratory, The University of Texas at Austin, January 1990.

[Venkatasubramanian and Rich, 1988] V. Venkatasubramanian and S. H. Rich. An object-oriented two-tier architecture for integrating compiled and deep-level knowledge for process diagnosis. *Computers and Chemical Engineering*, pages 903–922, 1988.

[Venkatasubramanian *et al.*, 1990] V. Venkatasubramanian, R. Vaidyanathan, and Y. Yamamoto. Process fault detection and diagnosis using neural networks—i. steady state processes. *Computers and Chemical Engineering*, 14(7):699–712, 1990.

[Vinson *et al.*, 1991] Jack Vinson, Stephan Grantham, and Lyle Ungar. Diagnosis using automatic rebuilding of qualitative models in chemical plants. Unpublished Article, supplied by Ben., 1991.

[Waterman, 1986] D. A Waterman. *A Guide to Expert Systems.* Addison Wesly, 1986.

[Weber, 1973] Thomas W. Weber. *An Introduction to Process Dynamics and Control.* John Wiley & Sons, 1973.

[Weld, 1988] Daniel Weld. *Theories of Comparative Analysis.* PhD thesis, MIT Artificial Intelligence Laboratory, 1988.

[Widman and Loparo, 1988] L. E. Widman and K. A. Loparo. *Artificial Intelligence, Simulation, and Modeling.* 1988.

Index

VITA

David Rutherford Throop was born in Akron, Ohio on October 27th, 1955, the son of Sara Thompson Throop and William Rutherford Throop. After completing his work at Walsh Jesuit High School, Northampton, Ohio in 1973, he entered Georgia Tech in Atlanta, Ga. He received a Bachelors of Science in Chemical Engineering in August 1989. He was employed as a chemical engineer for three years. In September 1983 he entered the Graduate School of the University of Texas.

Permanent address: Bx 7395
Austin, Texas 78713

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.