Earlier chapters discussed diagnostic systems based on classification models. We now extend the discussion to include plan-based models, causal network models, and detailed physical models for diagnosis.

9

Diagnosis and Troubleshooting

9.1 Introduction

To diagnose is to observe a physical artifact that is malfunctioning or a person who is ill and then to determine what is wrong through reasoning and further observations. The term *troubleshooting* refers to the diagnosis of physical devices such as mechanical and electronic systems. Troubleshooting is sometimes characterized as the diagnosis and repair of devices that have stopped working. Different domains and situations place different demands on diagnosis and troubleshooting.

9.1.1 Diagnosis and Troubleshooting Scenarios

The following scenarios illustrate some similarities and differences among diagnostic and troubleshooting tasks.

- *Infectious disease scenario.* A patient comes to a doctor complaining of listlessness, fever, and aching joints. The doctor takes the patient's temperature, listens to his breathing, and takes a culture sample while examining his throat. She determines that the patient has a bacterial infection and administers a broad-spectrum antibiotic. Later she checks the results of the throat culture to verify that the given antibiotic should be effective.
- *Board troubleshooting scenario.* A computer maintenance technician receives a complaint that the mouse on a computer workstation does not work. He replaces the mouse with a spare one, but the cursor on the display still does not track mouse movements. He then opens the keyboard case where the mouse plugs in, probes the output of an oscillator with an oscilloscope, and decides that the oscillator is working. He probes an interrupt line on

the output of a microprocessor chip and discovers that it remains stuck at five volts whether the mouse is stationary or moving. He probes several other points in the circuit and determines that one of the chips needs replacing. He turns off the system, replaces the chip, and tests the system again. The system works fine.

- *Picnic scenario.* Late on a hot afternoon of a big picnic, several people experience nausea, vomiting, muscular weakness, and disturbed vision. A physician and a member of the local health department come to the scene. The officials ask people what they ate, take samples of different foods, and run tests on the samples. The picnickers are treated for food poisoning. Shortly thereafter, another picnicker reports the same symptoms. The doctor learns that the picknicker ate some of the potato salad and treats him for food poisoning.
- *Communications box scenario.* A communications technician is notified that a communications switching controller is not working. She checks the lights on the front panels of the communications controllers and turns off the controller that has red lights glowing on the front panel. Inside the box are five identical circuit boards. She replaces the first board, turns on the unit, and determines from the red lights that the unit is still not working. She repeats this for the second board in the unit with the same result. She repeats it for the third board in the unit, and this time all the lights turn green and communications start working again. She places the original third board in a briefcase for broken boards and returns with it to her service van.
- *Copier troubleshooting scenario.* A reprographics technician receives a phone call that a high-speed copying machine is making illegible copies. He comes to the machine and tries several test patterns, verifying easily that the copy quality is below specifications. He plugs a portable computer into the machine and discovers that earlier that morning the machine's internal print-quality feedback loop adjusted several of its parameters to extreme values. He looks at the maintenance log and notices that the toner was also replenished that morning. He examines the toner powder and discovers that someone has filled the toner reservoir with the wrong kind of toner for the machine. He vacuums out the improper toner from the machine, refills the reservoir with the correct toner, and verifies that the copy quality is within specifications.

These scenarios have much in common. In each case, the diagnostician (doctor or technician) gathers and selects information about the malfunctioning system and its environment and uses the information to determine a treatment or repair.

9.1.2 Dimensions of Variation in Diagnostic Tasks

There are substantial differences in the above scenarios, illustrating important variations in diagnostic and troubleshooting tasks.

- *What can be a cause.* The scenarios differ in the nature of the causes of failure, such as broken components versus diseases versus environmental causes. In the board troubleshooting example and the communications box scenario, the technician's model of the malfunctioning system is expressed in terms of broken components. In that context, to diagnose a system is to determine which components need to be replaced. This contrasts with the infectious disease scenario, where the difficulty could be an externally introduced infection

that spreads throughout the patient's body. In the picnic scenario and copier scenario, the causes are neither diseases nor component failures. In the picnic scenario the picnickers consumed toxins from spoiled food; in the copier scenario the machine was given a toner powder that did not meet its operating requirements. These scenarios implicate causes in the environment.

- *Availability of intermediate diagnostic measurements.* The availability of diagnostic measurements varies in the scenarios. In the board troubleshooting example, a technician can probe any exposed wire in the system. To partition a diagnostic solution space efficiently, intermediate system data are needed. Because of the risk of complications, however, physicians prefer noninvasive techniques. Physicians usually have very limited options for obtaining or reasoning about internal data about a patient. Thus, there are differences across domains in the accessibility and usability of information about intermediate states in systems. Accessibility of state can change with improvements in instrumentation. Medical practice continues to become more sophisticated in its use of technology for imaging and measuring body function. At the same time, desire for large-scale integration and high density in manufacturing leads to compromises in provisions for accessing the internal states of electronic devices.
- *Specificity of diagnosis.* Another variation in these scenarios is the required specificity of diagnosis. In the communications box example, the technician only needed to determine which of five identical boards had failed. She did not care which component in the board was damaged. In the parlance of field service, a board is an example of a field replaceable unit (FRU) and a chip is not. Diagnosis in field work need not be more precise than the possible repair options. Similarly, in the infectious disease case, the doctor did not establish a precise vector of infection, but rather treated the patient with a broad-spectrum antibiotic.
- *Modifiability of system state.* The systems being diagnosed offer different opportunities for a diagnostician to put them into different states. In the board troubleshooting case, a diagnostician could have a special unit to arbitrarily drive the input of the board with different test vectors. This makes it relatively easy to put the system under diagnosis in a different state. In the copier scenario and medical scenarios, the systems (or people) have internal feedback loops that automatically compensate for changes in the external environment. Feedback loops make it more difficult to change the internal state.
- *Steady state versus dynamic behavior.* Time-dependent behavior can substantially complicate diagnosis. In medical systems, some diseases have periodic symptoms and some diseases are progressive. Medical diagnostic systems differ in terms of whether they treat each diagnostic session as an isolated event or whether they can reason about multiple successive visits of a patient. Electronic and troubleshooting systems typically carry out different functions over time. Some faulty behaviors occur only when certain functions are being performed. Diagnostic and troubleshooting systems differ in whether they model systems as having steady-state or time-varying behavior and in whether they can reason about intermittent faults.
- *Task overlap among diagnosis, testing, and treatment.* In many real applications, the useful tasks that people perform merge diagnosis with therapy or testing. In the communications box scenario, the technician replaces the boards one at a time until the controller works. In this case, observations of the operation of the controller before and after board replacement

provide data about the possible cause. Repairs and therapy can be tried experimentally to determine whether they are effective. In such cases, therapeutic and repair actions also yield data about the system and its faults.

Similarly, testing is the process of certifying that a system is fully operational. In a manufacturing setting, testing is a procedure for systematically checking the operation of a large practical subset of a system's behavior. In a medical context, the analogous task is a multiphasic physical exam. In contrast, diagnosis is usually undertaken to remedy an identified misbehavior and does not necessarily include looking for other problems not yet detected. Nonetheless, although diagnosis does not necessarily imply a thorough checking of all possible conditions and behaviors, it is often expected that certain obvious and perhaps simple tests be carried out so the repaired system will not immediately fail again, albeit due to a different cause. In many real situations, diagnosis, repair, and testing tasks are combined.

9.2 Models for Diagnosis Domains

In this section we develop a general model for knowledge-based diagnosis. We consider the basic phenomena and then compare approaches. In this way we seek to understand what diagnosis is, what makes it difficult, and what roles knowledge and various assumptions play in it.

Search Spaces for Diagnosis

So far the discussion has focused mainly on how diagnostic tasks can differ. Figure 9.1 presents a framework for describing what they can have in common. It shows the search spaces and operations that underlie knowledge-based systems for diagnosis. The three main spaces for diagnostic reasoning are a data space, a hypothesis space, and a therapy or repair space.

The **data space** presents a finite set of possible measurements, also called probe points, probes, observations, observables, manifestations, findings, history items, laboratory data, and signs and symptoms. For diagnostic reasoning, we distinguish different kinds of values for each data variable, such as possible values, normal values, predicted values, and observed values. The possible values are the ones that can be considered at all in a given diagnostic model. Predicted values are those expected from a model and from what is known. Observed values are those reported by instruments coupled to the system in its physical environment. Normal values are those that can occur within the normal operating conditions of the device being modeled.

The **hypothesis space** represents diagnostic hypotheses about what could be wrong with the system being diagnosed. A hypothesis can involve multiple underlying faults, also called syndromes, diseases, or causes, depending on the domain. The **therapy or repair space** represents a set of possible actions to treat or repair the system.

Below all three spaces in Figure 9.1 is another box labeled "**System Model**." Roughly, a system model describes the structure and behavior of the system being diagnosed, including some elements of its environment. The three spaces and the system model reference many of the same representations and may usefully share common procedures and data structures in their implementations. For example, it is useful to be able to install hypothetical faults in the system model corresponding to faults postulated by diagnostic hypotheses and to predict resulting values in the data space. Similarly, we can install various repairs in a model and then predict the result-

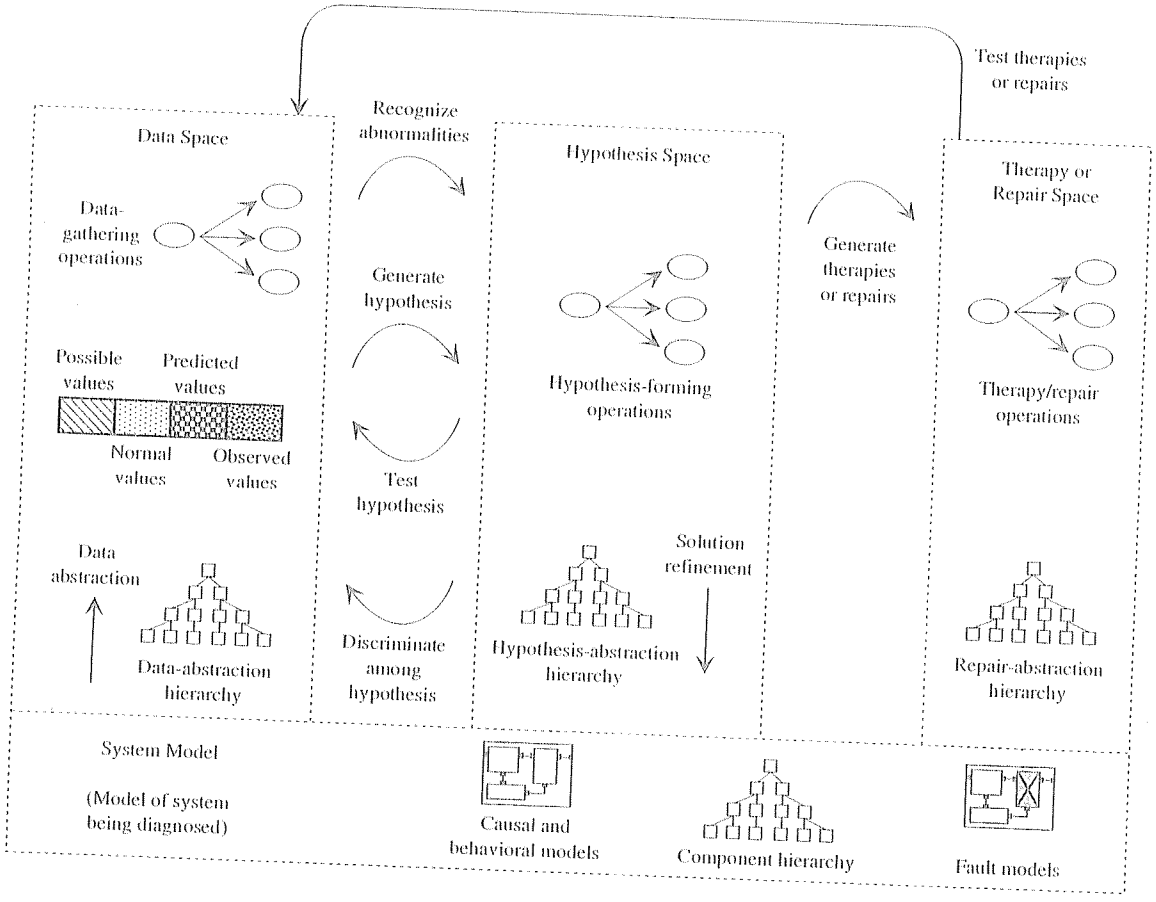


FIGURE 9.1. Search spaces and shared system models for diagnosis tasks.

ing changes in system behavior. In the following, we discuss the spaces and system model further to develop an informal definition of diagnosis.

All three spaces describe operations. Data-gathering operations carry out observations of the system under consideration. Therapy and repair operations in the therapy and repair space make changes to the system to fix the problem. The central operations in diagnosis tasks are the ones that manipulate diagnostic hypotheses. The operations in the hypothesis space include generating hypotheses and discriminating among competing hypotheses. These operations are carried out in recurring patterns of search and knowledge use. These patterns exemplify common elements characteristic of knowledge-based diagnostic systems.

All three spaces can have representations at multiple levels of abstraction. As in the horseshoe diagram for classification, a data space can include data abstractions that simplify mapping to fault or disease hypotheses. A qualitative abstraction may map a particular measurement of body temperature to "fever." Diagnostic hypotheses can be organized in terms of general and specific hypotheses. For example, the hypothesis that there is a fault in the "adder" is more general than the hypothesis that there is a "stuck at 1" fault in a particular chip that is part of the adder.

An Example of a System to be Diagnosed

Figure 9.2 gives an example of a simple (albeit fictional) system to be diagnosed, TS-1, that we use to illustrate concepts and issues in the following. In this example, the components A1 and A2 are hardware adders and M1 and M2 are multipliers. The data space in this example includes descriptions of values for the inputs and outputs of each component. For example, M1 has the values 3 and 2 on its inputs. Measurements can be taken on the input and output points for the system and also on the internal test points T1, T2, and T3.

A behavioral model indicates what the different components do. In our simple model, the adders add their inputs and deliver the results on their outputs. The multipliers multiply their inputs together and deliver the results on their outputs. Adders and multipliers have limited precision. In TS-1 the components use a 5-bit binary representation for positive integers. This representation accommodates numbers between 0 and 31, where any bits representing larger numbers are simply lost. Figure 9.3 presents behavior rules for the adders and multipliers in our example.

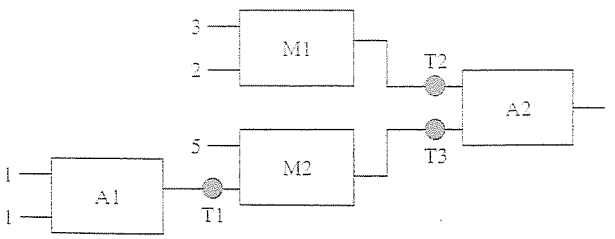


FIGURE 9.2. TS-1: An example of a system to be diagnosed, composed of two multipliers and two adders.

FIGURE 9.1. Search spaces and shared system models for diagnosis tasks.

Behavior Modes for Adder

Mode	*Probability*	*Behavior*
w (working)	.9990	Out = (In1 + In2) mod 32
s (shorted)	.0003	Out = 0
d (drop MSB)	.0006	Out = (In1 + In2) mod 16
u (unknown)	.0001	—

Behavior Modes for Multiplier

Mode	*Probability*	*Behavior*
w (working)	.9990	Out = (In1 * In2) mod 32
s (shorted)	.0003	Out = 0
d (drop MSB)	.0006	Out = (In1 * In2) mod 16
u (unknown)	.0001	—

FIGURE 9.3. The behavioral modes of the components in our sample model, including the normal working mode, known fault modes, and a catchall unknown fault mode. The probabilities provided by the manufacturer are the prior probabilities that a given component is broken. The abbreviation MSB refers to the most significant bit.

The names In1 and In2 refer to the two inputs to the adders and multipliers. We use the convention in our circuit diagrams that In1 is the upper input and In2 is the lower one.

In Figure 9.1 the behavior and component specifications are both part of the system model. The other major part of the system model is the fault model, where a "fault" is what can go wrong in TS-1. As discussed earlier, fault models can include diseases in medical diagnosis systems or environmental effects. In TS-1, the fault model expresses how components can be broken. It specifies the behavior of specific kinds of faults and gives the prior probability that a component of the given type is faulted in that way. In practice, we assume there is a finite (and usually quite small) number of possible faults. Probabilities of faults can be determined by statistical testing, provided by the fictional manufacturer of components. Figure 9.3 lists the various behavior modes of the adder and multiplier in TS-1, including the normal working mode and a catchall unknown fault mode for both kinds of components.

The Main Subtasks for Diagnosis

The basic paradigm for diagnosis can be understood as the interaction of observation and prediction. Observation is based on what we can see or detect about a physical artifact, the system of interest. Prediction is based on a computational model of the system, which indicates what it is supposed to do under various circumstances. The model supports expectations about the system's behavior. When a difference or discrepancy is detected between between observations and predictions, we begin diagnosis. Presuming that its model is correct, a diagnostic system attempts to account for the discrepancy in terms of defects in the physical system, such as broken components or, in the case of medical diagnoses, diseases of various kinds.

The main subtasks in diagnosis are:

- *Recognizing abnormalities and conflicts.* A diagnosis task usually starts when some data are presented indicating potentially abnormal symptoms or measurements. The first step in

diagnosis is to determine how the data acquired so far conflict with expectations. This step identifies ways in which the data are abnormal, are inconsistent with each other, or are inconsistent with a model of the system being diagnosed.

- *Generating and testing hypotheses*. The next step is to identify or generate hypotheses to explain the data conflicts. In a hierarchical approach, this step may involve refining general hypotheses. Testing means to determine which hypotheses are consistent with the data, which are inconsistent, and which are unresolved. In principle, generation could be a simple and systematic process for constructing alternatives, to be followed by a separate process for knowledge-based testing and ranking of the alternatives. In practice, however, diagnostic systems use similar knowledge to guide both.
- *Discriminating among hypotheses*. This step reflects the goal of making a sharp diagnosis. It must gather new information about the system, such as by selecting a new probe point, by running tests, or by perturbing the system state and gathering new data. It determines what new information is needed to rule out hypotheses.

These three subtasks are our points of departure for comparing different approaches to building knowledge-based systems for diagnosis. For each subtask we explore the phenomena that arise and compare approaches.

Therapy or repair selection involves considerations beyond those for diagnosis itself, especially in cases where there are long-term treatments or progressive diseases. Analysis of the therapy or repair space and its associated operations is mostly outside the scope of this chapter. Chapter 8 includes a case study of the therapy model used by MYCIN, based on a configuration model.

9.2.1 *Recognizing Abnormalities and Conflicts*

In diagnostic practice, some evidence is used initially to determine whether to perform diagnosis. In a medical setting, the initial data are called the "presenting symptoms." In some domains the determination of whether the data indicate an abnormal condition is just a simple screening step preliminary to the main parts of diagnosis. Other diagnostic systems always perform a diagnosis, starting with whatever data the system is given. In the following we consider some of the issues that arise.

The task at this point is to determine whether the known data about the system indicate that the system is in an abnormal state, requiring diagnosis and subsequent repair or treatment. This is a smaller job than diagnosis. We do not need to determine exactly what is wrong. We need to decide whether to begin a thorough diagnostic process. This task includes making judgments about the signal or probe values at hand. For the present, we ignore systems that adjust themselves, such as self-adjusting equipment or homeostatic biological systems. In such systems, determining whether the system is in a "normal" state can be complex.

Figure 9.4 presents an example where an abnormality or conflict would be recognized for TS-1. In this example, we are given all the integer inputs to the circuit. The dashed boxes indicate predictions made using the behavior rules for TS-1. For each prediction, the components listed between braces are assumed to be working. In other words, the rule used to predict the value assumes that the component is in the normal working behavior mode. A conflict arises when the output of A2 is predicted to be 16 but is measured to be 0.

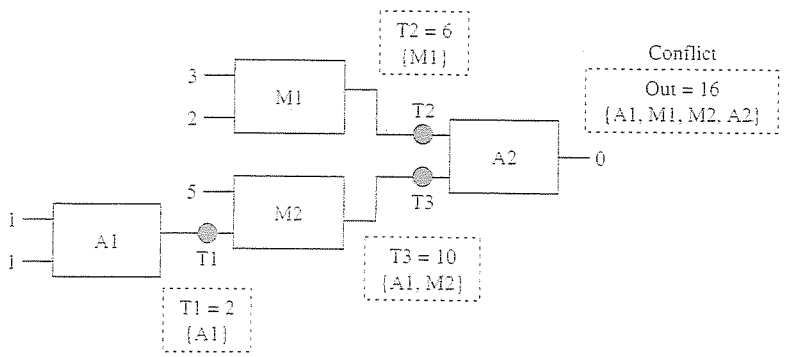


FIGURE 9.4. Example of a conflict or abnormality noted in TS-1, indicating that some measured value is inconsistent with other measured values and the assumption that the components are all working. The numbers given next to inputs and outputs in the diagram are measured probe values. For example, the inputs to A1 are both 1. The values given in dashed boxes are values predicted by the system model assuming that the listed components are working.

Figure 9.5 presents a functional diagram of another, more complex system to be diagnosed. The figure is a block diagram for an electronic circuit, but in principle it could represent any kind of a system. On the left side of Figure 9.5 are shown a number of causes, input signals or "inputs." The question marks indicate signals whose values are not known. Effects or outputs of the system are also shown. These are the labeled lines on the right side of the figure. For example, the value of the output signal *a* is not known and the value of the output *b* is 1. In some systems, the distinction between inputs and outputs is not meaningful and the terminology of causes

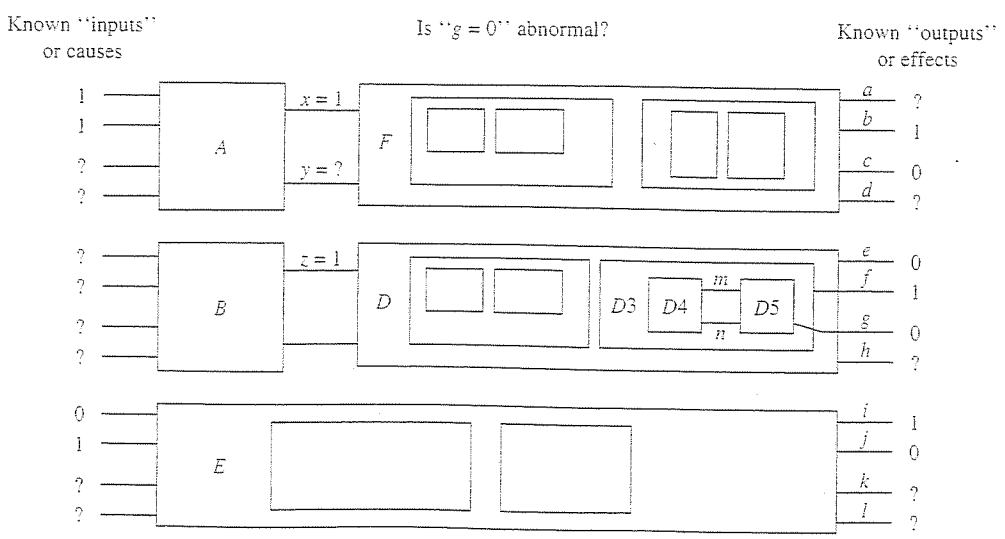


FIGURE 9.5. Functional diagram of a system being diagnosed.

and effects is preferred. In a medical example, "heart rate," "temperature," and "nausea" might be examples of effects.

Given that the output g has value 0, we ask whether this value is normal or expected. Various properties of the system in Figure 9.5 can conspire to make this question less straightforward than it might seem. If we have a complete simulation model of the system, then in principle we can run the simulation given the inputs, predict the outputs or effects, and compare them to what is measured. Wherever the predictions are different from what has been measured, we note the conflicts.

This approach is not always feasible because it requires knowing so much. The process of computing a value for signals like g requires that we know values for other signals that may not be available. For example, if the value of g depends on the unknown values for the input signals to module B, then the simulation value for g cannot be determined.

Because of the clocking of signals and devices in digital circuits, a signal can usually flow only from inputs to outputs. An adder cannot be made to subtract by reversing the "flow" of its signals. However, this physical directionality is not a limitation for constraint propagation in diagnostic reasoning. Suppose we are given that the output of an adder in TS-1 has the value 10 and that one of its inputs has the value 6. Using its addition behavior in "reverse," we can subtract and predict that the value of its other input is 4. Rules for reasoning backward from component output values (effects) to input values (causes) are sometimes called **antibehavior rules**. Using such rules increases the number of predictions that can be made from limited data. In either case, predictions are of the form "g will have the value 1 if $f = 1$ and $D4$ and $D5$ are working." Contradictions have the form "given that $g = 0$ and $f = 1$, at least one of the components $\{D3, D4, D5\}$ must be broken."

Most approaches to prediction are logically incomplete. This leads to cases where additional analysis can be used in making predictions and in finding contradictions. For example, it might turn out that $g = 1$ for *all* possible combinations of inputs to module $D5$. In such cases, we can infer the output values even when the particular input values are not known. Given no values for signals m or n, one constraint propagation scheme might be stymied where another could determine that $g = 0$ is always abnormal, implicating components inside $D5$.

Figure 9.6 summarizes approaches for determining whether diagnosis is needed. These approaches differ in the form of analysis they carry out. On the right side are approaches that use system models to predict behavior using physical models and the assumption that components operate according to known rules.

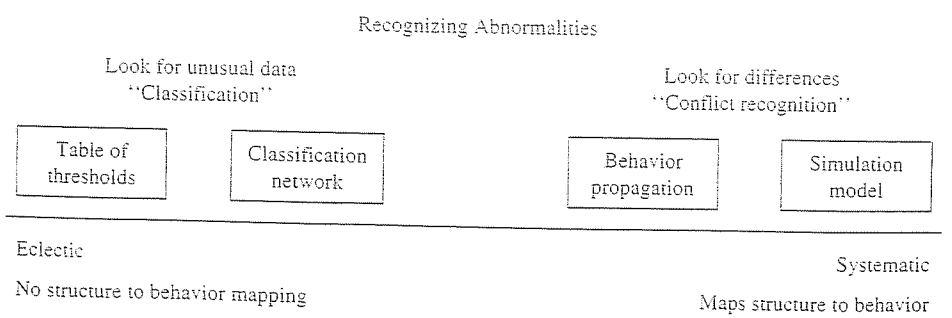


FIGURE 9.6. Alternative approaches for recognizing abnormalities and beginning diagnosis.

On the left side are classification approaches that classify the data directly as normal or abnormal. A classification approach can be eclectic. Rules for identifying particular cases of abnormal data, which may be important but overlooked by systematic but incomplete constraint propagation strategies, can be added on a case-by-case basis. Classification approaches to detecting abnormalities can be implemented equivalently as decision networks or rule systems. In simple cases where the data are independent, classification may simply use separate tables for each signal.

9.2.2 *Generating and Testing Hypotheses*

A diagnostic hypothesis is an account of the causes of symptoms. Hypothesis generation involves reasoning from symptoms to diagnoses, which are explanations of what may plausibly have caused the symptoms. Hypothesis generation is a search problem, and there are many ways to approach it. In the ideal case, the search process tries to generate all possible diagnostic hypotheses. In practice, search methods prune low-probability diagnoses early.

To test a hypothesis is to compare predictions made on the basis of the hypothesis with observed data. When the predictions for a hypothesis do not bear out, the hypothesis is usually rejected. Because the reasoning and representation issues are so intertwined, we consider generation and testing together in the following.

Several themes are woven into our discussion of hypothesis generation and testing:

- *Open-ended models of failure*. The simplest troubleshooting systems describe diagnoses in terms of specific kinds of component failures. But failures can result from many different kinds of interactions, some of which might not be included in a particular model. More powerful approaches to modeling expand the basis for plausible diagnoses incrementally in a systematic way.
- *Singleton versus composite diagnoses*. The simplest diagnostic systems assume that the system being diagnosed has only a single fault or a single disease. This assumption greatly simplified the earliest diagnostic systems and is still widely used. Unfortunately, it is not generally valid and leads to incorrect results. A second approach is to assume that faults can be determined one at a time and then subtract away the corresponding symptoms, analyzing the "residue" for further diagnoses. This approach also leads to certain systematic difficulties. Better approaches use composite diagnoses, in which different combinations of possible faults or diseases are considered as a group. The challenge in this approach is to control the combinatorics.
- *Hypothesis hierarchies*. The combinatorics of composite diagnoses and other considerations of efficiency lead to approaches to diagnosis based on hierarchical search. Typically, the hierarchies are based on abstraction and containment relations of functional subsystems.
- *Independent versus interacting causes*. A basic issue in diagnosis is determining which evidence implicates which faults or diseases. Systems lacking models of how symptoms interact have inherent difficulties in assigning credit or blame to diagnoses.

We now consider these themes in more detail.

Models of Subsystem Failures and Interactions

The data space is about what is observable and the hypothesis space is about what is conceivable, meaning what can conceivably go wrong with a system. In the following we consider the variety of approaches that have been used in characterizing this. The most basic issue in defining a diagnostic hypothesis space is determining and specifying what can go wrong in the functioning of a physical artifact.

In the simplest troubleshooting systems a diagnosis is expressed in terms of faults of particular components. In TS-1, each component has four behavior modes. Besides the normal mode, there are three abnormal modes including one mode where the behavior of a component is not known. Diagnostic systems generally have some most primitive level of description for components. The space of possible diagnoses is determined by accounting for all the possible faults for the primitive devices. Thus, in TS-1 where adders have four possible modes, then either adder could be in any of these modes.

In the design of component models, much variability is possible not only in the level of description but also in deciding what constitutes a part and what faults are possible. Even in electronic domains, where there is a great deal of engineering practice to draw on, variations are possible. We illustrate this by considering the role of "wires" in models for electronic troubleshooting.

In many troubleshooting systems, a wire is not modeled as a device and is therefore not recognized as being capable of having a fault. This presents no difficulty so long as no "wire faults" occur in practice. However, in all electronic technologies, wires are conductors and are subject to faults. If a conductor gets too thin, its electrical resistance increases. If it is thin at a particular point or if too much current passes through a wire, it can melt, leading to an open circuit. Conductors are separated from other conductors by insulating material. Extraneous pieces of conductor material or holes in the insulators can result in unintended conductive connections. These shorts are called "bridging faults." If we want to predict whether a bridging fault is likely between two wires, then we need to take into account the physical layout and manufacturing techniques. In general, the likelihood of a bridging fault is greater if two wires are close together. There can also be interactions between conductors even when they are not connected. When current flows through a wire, it also sets up electrostatic and electromagnetic fields. If two wires are near each other, these fields can cause capacitative or inductive coupling of signals between the wires even if they do not touch. Such coupling usually corresponds to a design error but can also result when long cables are unintentionally moved close to power lines.

The point of this discussion about wires is that the "fault possibilities" that we build into our models reflect assumptions about what physical effects in the system are likely and important. Behavior depends on causal pathways, by which activity on one part can have effects on another. To answer the question "what can go wrong" in a system, we need to determine what mechanisms are present and what parts of a system can interact. In the process of designing an artifact, certain causal pathways are emphasized over others. Although the models we used in our diagnostic systems tend to emphasize the interactions that are planned in the design, there may be other interactions, too.

The notion of causal pathways also depends on identifying what interactions between components are possible. A diagnostic system explains malfunctions using only the terms and phenomena in its model. When some pathways of interaction are not included, the diagnostic system

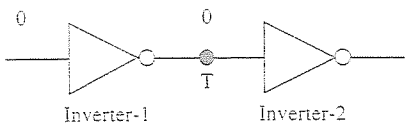


FIGURE 9.7. If test point T is not 1, the determination of suspected components depends on which interactions are modeled. In the most common electrical model, only Inverter-1 is suspected. In a model that includes interactions through current flow, a failure of either inverter could cause the effect. (Adapted from an example in Hamscher, 1988, page 30.)

is blind to some of the possibilities. For example, Figure 9.7 shows a simple electronic circuit with two inverters in series. A common model of digital circuits says each node is driven to 0 or 1 by just one gate. According to this model, a discrepancy at testpoint T can only be caused by nodes upstream, that is, by Inverter-1. However, the gates also interact through current flow. It is possible for the second inverter to fail in such a way that it pulls down the voltage at the test point. In this case, the common digital model yields the wrong answer because it does not include a possible pathway of interaction.

Diagnostic systems make closed-world assumptions about the range of possible interactions. The issue here is not that models make assumptions. Indeed, simplifying assumptions are essential to keep models tractable. The issue is that the assumptions need to be identified and managed.

Recognizing the tension between the completeness or precision of a model and the need to constrain the diagnostic search, Davis and Hamscher (1988) proposed that candidate generation should be thought of in terms of causal pathways of decreasing probability. Experienced engineers know that some things are more likely to go wrong than others. As a result, they first attempt to generate solutions that employ simpler and more likely hypotheses. Only in the face of a contradiction do they fall back on more elaborate and less likely possibilities. Davis and Hamscher found three principles in the cognitive models used by engineers:

1. Engineers have a set of categories of what can go wrong.
2. There is an ordering criterion that indicates which categories of hypotheses to entertain first.
3. The categories are ordered but none is permanently excluded.

These principles embody an alternative to a fixed closed-world approach in the spirit of the concentric models of exceptions discussed for the qualification problem in Chapter 3. At each stage, there is a search space of possible diagnoses. When nothing in the space amounts to a satisfactory solution, the space is expanded to include categories previously thought too implausible. Metaphorically, a single closed world is replaced by a series of concentric worlds, each representing more pathways for interactions than its predecessor. Within each concentric world it is important to keep the search for hypotheses as constrained as possible. In the bridge-fault example, a system should not examine every pair of adjacent pins, but just the ones that could account for the observed discrepancies.

Diagnoses: Single Faults versus Multiple Faults

The most likely cause of misbehavior in many systems is the failure of a single component. Jumping from plausibility to certainty, many simple diagnostic systems assume in every case that there is at most one fault or disease. This simplifies the generation of hypotheses and enables reasoning by elimination.

Unfortunately, the single-fault assumption is not always valid in practice. Not only is it possible for multiple faults or diseases to co-occur, but also they are not always independent. For example, a failure of one electronic part may cause other parts of a circuit to become overloaded and subsequently to fail. Bad alignment of wheels in a car can cause tires to wear badly and to fail. A patient weakened by one disease is often more vulnerable to additional afflictions.

To simplify the accounting of multiple causes, some diagnostic systems adopt a sequential, or one-cause-at-a-time, approach. The basic idea is to order the symptoms in terms of importance and to iteratively identify which candidates explain the main symptoms. When the best-matching causes are selected, their symptoms are marked as "accounted for" and causes for the "residual symptoms" are then sought. This approach follows the ground rules of sequential decision making in which one decision problem must be resolved before its successor problems are started. Unfortunately, the complications of interactions among causes as discussed later make this approach unreliable.

An attractive alternative is to consider combinations of multiple faults at the same time. This is called a **set-covering** or **composite-diagnosis approach**. This technique goes back at least to Reggia, Nau, and Wang (1983). It was one of the most important advances in diagnostic systems over the past few years because it provides a systematic basis for reasoning about difficult diagnostic cases, where there are separate but potentially interacting effects of multiple faults or diseases.

The main advantage of this approach is that it provides a systematic basis for considering the interacting effects of multiple causes. The main challenge with the approach is that there are so many combinations of causes! Every combination of faults is a possible diagnostic hypothesis. The smallest set of components is the empty set, meaning there are no broken components. When the single-fault assumption common to many diagnostic programs is used, there are only n candidates involving one component each for a system of n components. However, in the general case, every component could either be working or not working, leading to 2^n combinations. Figure 9.8 illustrates a graph of these combinations, arranged according to the terms of the binomial theorem. The bottom row contains all combinations of single faults, the next row all combinations of double faults, the next row all combinations of triple faults, and so on. The lines connect sets in each row to their supersets in the next row above.

As shown in Figure 9.9, there are 32 combinations of faults for a system with 5 components. This exponential complexity is inherent in any approach that can consider all possible combinations of faults and is a major obstacle in realistic diagnostic problems. Even considering only one fault mode per component, there are a total of 2^{50}, or roughly 10^{15}, possible candidates in a 50-component system. The probability of individual candidates usually drops off rapidly as more faults are postulated. Nonetheless, a best-first search in a 50-component problem would contend with roughly 20,000 candidates of up to three faults. Suppose we consider larger and more realistic electromechanical systems of 500 components. As Figure 9.9 indicates, even limiting attention to candidates with no more than three faulty components, there are millions of can-

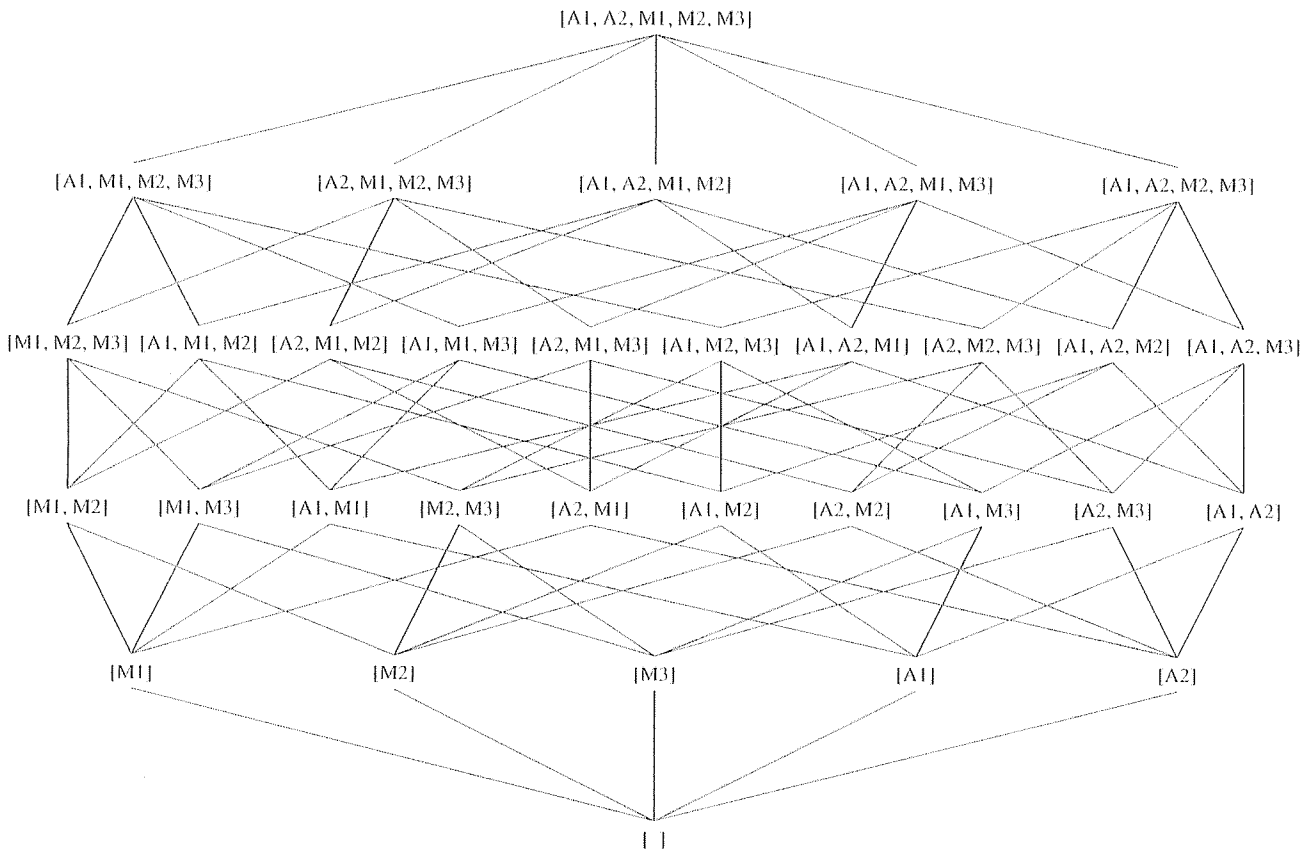


FIGURE 9.8. Sets and subsets of the five components, A1, A2, M1, M2, and M3. Each set corresponds to a diagnostic hypothesis, where the elements in the set are faulty and other components are working. The top row is the set where all components are broken and the bottom row is the set where no components are broken. Each row corresponds to a term in the binomial theorem, indicating the combinations of k broken components. Thus, there are 5 single-component sets, 10 double-component sets, 10 triple-component sets, 5 quadruple-component sets, and 1 set of 5 components.

Number of Components	*Number of Possible Diagnostic Hypotheses for k faults*			
	k=1	*k=2*	*k=3*	*Total Over All k*
5	5	10	10	$2^5 = 32$
10	10	45	120	$2^{10}=10^{24}$
50	50	1,225	19,600	$2^{50}=1.1 \times 10^{15}$
100	100	4,950	161,700	$2^{100}=1.3 \times 10^{30}$
300	300	44,850	4,455,100	$2^{300}=2 \times 10^{90}$
500	500	124,750	20,708,499	$2^{500}=3.3 \times 10^{150}$
1,000	1,000	499,500	166,167,000	$2^{1000}=1.1 \times 10^{301}$
5,000	5,000	12,497,500	20,820,835,000	$2^{5000} = 1.6 \times 10^{1505}$

FIGURE 9.9. The number of candidate diagnostic hypotheses grows exponentially.

didates to consider. A goal of researchers working in this area is to diagnose systems with thousands of components. In the following sections we see how different system models affect diagnostic complexity and how different approaches seek to cope with it. Every diagnostic system that can reason about multiple faults needs to cope with this explosion by drastically narrowing the set of candidates.

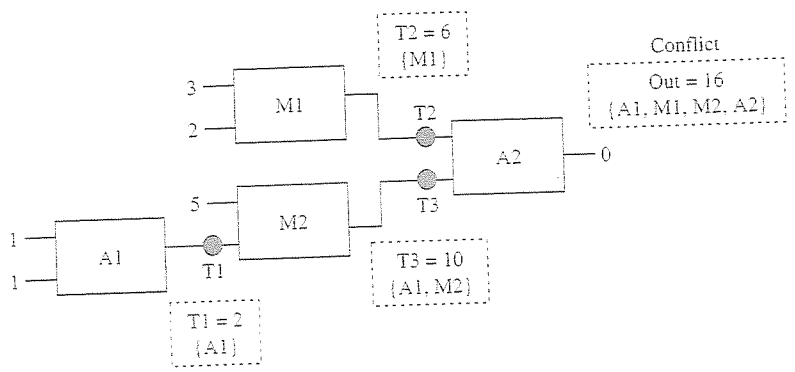
You may wonder at this point whether handling multiple faults is important enough to warrant all this complexity. The importance of multiple faults depends a great deal on the situation. In medicine, the care of indigents often involves treating multiple infections and other problems. In electronic and mechanical systems, the failure of one component can cause other components to fail. In cases where reliability is a crucial or where troubleshooting and repair are expensive or inconvenient, an ability to diagnose and repair multiple faults has substantial value.

Figure 9.10 returns to our TS-1 model to present an example of computing candidates and their probabilities. In this figure we have the conflict or abnormal reading at the output of A2 as shown. One explanation for the conflict is that the component A2 is in mode *d*, meaning it has dropped its most significant bit. In the figure, this diagnostic hypothesis is indicated by the notation $[A2_d]$, which means A2 is in the *d* mode and all the other unnamed components are in working modes. The prior probability of this singleton candidate is the product of the probabilities that each component is in the mode indicated.

Figure 9.11 gives an example of a possible composite candidate, $[M1_s, M2_s]$, explaining the same conflict. This diagnostic hypothesis is that components M1 and M2 are both shorted, so that their outputs are zero. Figure 9.11 gives the prior probability of this candidate in the same fashion as before. From the example it is clear how the prior probability of a composite candidate is usually much less than the prior probability of a singleton candidate.

Hierarchies for Diagnostic Hypotheses

Hierarchical approaches reduce the combinatorics of search on large problems. The design and roles of hierarchies in diagnosis domains relate not only to hypothesis generation, but also to hypothesis testing and discrimination. As in hierarchical approaches to other kinds of problem solving, hierarchies provide a means of describing and reasoning about classes of solutions rather than just individual solutions. The effectiveness of a hierarchical approach depends on the



Candidate $[A2_d]$

A2 is in mode *d*. A1 is in mode *w*.
M1 is in mode *w*. M2 is in mode *w*.

Prior Probability

$p([A2_d]) = .0006 * .9990 * .9990 * .9990 = 5.982\ 10^{-3}$

FIGURE 9.10. Example of a singleton candidate and its prior probability.

availability of knowledge and problem-solving operations that can exploit the class boundaries induced by the hierarchies. As usual, the strategy of a hierarchical approach is to divide and conquer.

Figure 9.12 presents a hierarchy for guiding hypothesis generation on a particularly simple system model. In case 1 in the figure, we begin with a single element, corresponding (say) to a single device like others in a system to be diagnosed. The behavior of element *A* is described relative to its four external connections. Case 2 connects 16 elements together. The resulting system has 16 external connections and 24 internal ones. Reasoning about the aggregate behavior requires that we compute the interactions of elements through all the viable combinations of states.

Case 3 suggests how four composed elements can be abstracted to a single "macro-element." In creating this abstraction we combine elements and connections together describing their aggregate behavior as a single element, but in less detail. Macro-connections are analogous to the aggregation of wires into buses that carry composite signals. In case 4 we compose four macro-elements together. Case 4 has the same number of primitive elements as case 2 but presents two smaller problems for diagnosis instead of one large problem. Using just one level of

Candidate $[M1_s, M2_s]$
M1 is in mode *s*.
M2 is in mode *s*.
A1 is in mode *w*.
A2 is in mode *w*.

Prior probability
$p([M1_s, M2_s]) = .0003 * .0003 * .9990 * .9990 = 8.982\ 10^{-8}$

FIGURE 9.11. Example of a composite candidate and its prior probability.

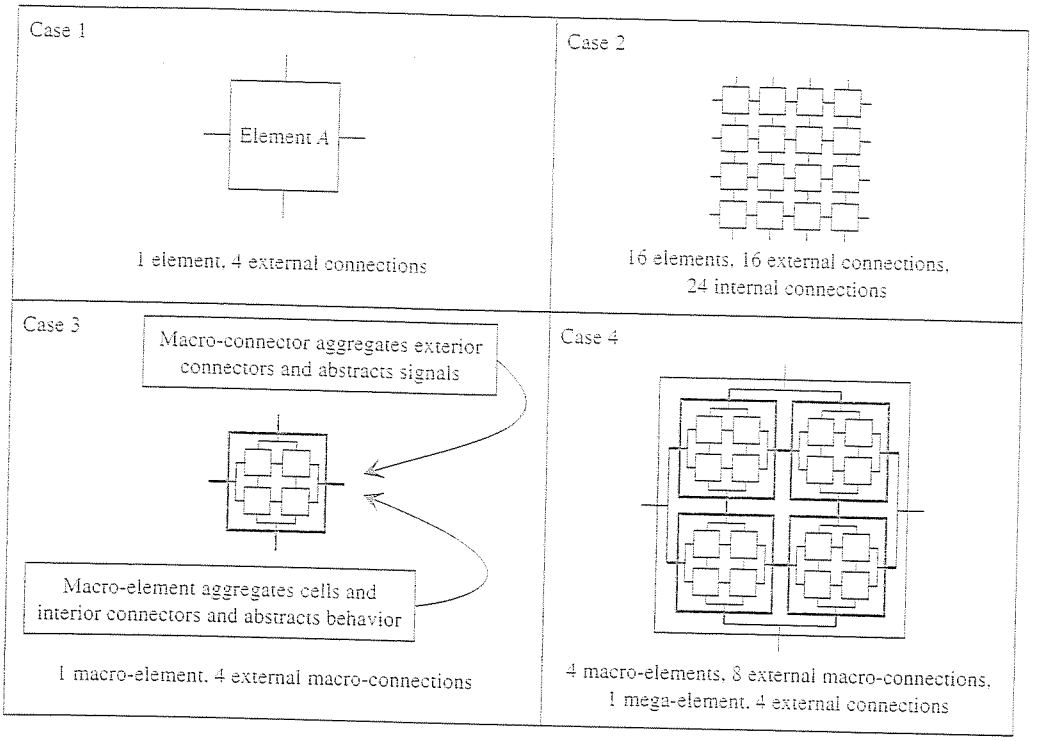


FIGURE 9.12. Abstraction is the key to reducing complexity in hypothesis hierarachies. Case 4 reduces the number of elements to be considered at once.

abstraction, the complexity of the outer description is reduced to four macro-elements and eight macro-connections. Recursively, we can introduce another "mega" level of aggregation and abstraction to bring us back to one mega-element with four mega-connections. Introducing multiple levels of abstraction transforms diagnosis to a sequence of smaller subproblems. In the exercises we consider how the time complexity of diagnosis depends on of the number and size of the subproblems.

Different kinds of elements and heterogenous mixtures of elements compose according to different rules. Typically, new abstractions are required for each level of aggregation. Still, the basic point about the combinatorics remains the same. The complexity of diagnosis can increase exponentially with the number of components. Abstractions extend the size of the systems that can be diagnosed. Most diagnostic practice uses **lumped-element models**, which are models that "lump" functionality into specific and separate elements rather than treating functionality as completely distributed in space. Describing such models requires abstracting both the elements and their connections.

All the hierarchical systems we considered have a bottom, most primitive level of representation. Higher levels of organization are defined by composition of the lower ones. Choosing a primitive level leads to three fundamental commitments. First, it makes some failures indistinguishable to the troubleshooting system. Every failure inside a primitive component will result in

the same diagnosis. Second, it makes some failures representable only as failures in multiple components. For example, a short circuit between two supposedly noninteracting components would be diagnosed as a failure of the two components. Third, the lower the level of detail, the more work is involved in predicting behavior. This is not to say that one cannot add parts or even levels later, corresponding to building a new foundation or basement under an existing house. In some practical systems where size is an issue or where a mixture of technologies has been used, it may be appropriate to represent different parts of the system at different levels of detail.

The use of hierarchies to organize levels of hypotheses is not limited to diagnosis of human-made systems. In medical diagnoses, concepts about organs and functional subsystems such as digestive systems, immune systems, and circulatory systems are often used in organizing diagnostic hypotheses. The value of this is easy to see. Organs and other subsystems tend to carry out a single major bodily function: transporting materials in the blood, removing waste products, exchanging oxygen and carbon dioxide, destroying foreign substances, and so on. Any disease of that organ tends to diminish its effectiveness in carrying out that function, causing similar effects and symptoms.

This phenomenon also arises in digital subsystems. Some behaviorally complex digital components can have many different kinds of internal faults. At any particular level of behavioral abstraction, some set of faults will collapse onto a single misbehavior. In hardware design, current design practice tends to minimize the number of parts and to implement complex behavior using software or firmware for microprocessors, which use the same hardware components repeatedly for several steps in an overall computation. In these implementations, failures at one stage tend to cascade so that complex devices soon fail catastrophically. Thus, after testing subsystems on simple cases, experienced troubleshooters can reliably treat entire complex subsystems as very low likelihood suspects. Many complex systems either work completely or fail utterly.

Finally, hierarchies need not necessarily be based on models of physical faults. In medical domains the hierarchies can taxonomize infecting agents in terms of families of related microorganisms, microorganisms that respond to the same treatments, or microorganisms that result in some of the same symptoms. A taxonomy of diseases in medicine is also called a nosology. There is no single right way to construct a **nosology**. Such taxonomies can be based on anatomical locus such as diseases of the liver, caustative agent such as type of bacteria, or symptoms such as swellings. Multiple relations for categorizing can be combined in a single framework of specialization. To cite an example from ABEL (Patil, 1981), renal disease is defined to be a disease of the renal system (anatomical locus). Renal failure is then defined as a renal disease characterized by low urine output (physiological locus). Acute renal failure is renal failure with sudden and severe onset, and, finally, drug-induced acute renal failure is acute renal failure of chemically toxic etiology. Each step in the above definitions defines a disease that is further specialized by one of its primary characterizations.

Independent versus Interacting Causes

Accounting for which symptoms correspond to which diagnoses is crucial, in both generating and testing diagnostic hypotheses. Figure 9.13 summarizes some possibilities both for the ideal case and for complications. In this figure, C1 and C2 are two possible causes of misbehavior in a system being diagnosed. The function *S* is the set of symptoms that are visible given a set of causes.

Ideal case

- Single symptoms, $|S(\{C1\})| = 1$. Each cause has exactly one symptom.
- No overlap, $S(\{C1\}) \cap S(\{C2\}) = \emptyset$. Different causes have no symptoms in common.
- Independence of symptoms, $S(\{C1, C2\}) = S(\{C1\}) \cup S(\{C2\})$. The symptoms observed when there are multiple causes is the union of the symptoms that are seen when the causes are present individually.

Complications

- Multiple symptoms, $|S(\{C1\})| > 1$. A fault or disease can have more than one symptom.
- Overlapping symptoms, $S(\{C1\}) \cap S(\{C2\}) \neq \emptyset$. Different faults or diseases can lead to the same symptoms.
- Compensating symptoms, $S(\{C1, C2\}) \subset S(\{C1\}) \cup S(\{C2\})$. When two diseases or faults are both present, *fewer* symptoms may appear than in the union of their separate sets of symptoms because of compensating effects.
- Synergistic symptoms, $S(\{C1\}) \cup S(\{C2\}) \subset S(\{C1, C2\})$. When two diseases or faults are both present, there can be *more* symptoms than there are in the union of the sets of their individual symptoms.

FIGURE 9.13. Idealized and realistic relations between causes and symptoms. In this figure, C1 and C2 are two possible causes of faulty behavior and S is a function that maps a set of causes to the set of observable symptoms.

The first phenomenon, multiple symptoms, is no surprise. Diseases and component faults can change several aspects of a system's behavior, so it is not surprising that a routine sampling of observations often detects more than one symptom from a single disease. However, this has implications for diagnostic systems that implicate causes from the presence of symptoms. One issue is that different symptoms are not necessarily independent. When they are related to or caused by the same underlying disease state, they should not be counted as independent evidence. Approaches that add "evoking strengths" cumulatively and independently from whatever symptoms have been observed favor diagnoses with the most commonly observed findings.

The second phenomenon, overlapping symptoms, is also familiar in medical domains because some symptoms are common to multiple diseases. For example, fever is associated with most types of infections, many forms of cancer, blood abnormalities, cardiovascular disorders, and diseases associated with abnormal immune systems. One approach to accounting for overlapping symptoms is to represent and reason about intermediate pathophysiological states. These intermediate states represent the fact that disease states both share symptoms and lead to multiple observables. This representation goes beyond the metaphor of having fixed fingerprints or "symptom profiles" for different diseases. It provides a basis for factoring symptoms into interdependent sets. It enables a diagnostic program to attribute symptoms to causes while diminishing the influence of interdependent manifestations of a single disease.

The compensating symptom phenomenon occurs when two diseases exert opposite influences on some observable parameter. For example, electrolyte and acid-base systems in the human body are influenced by many factors. A patient with salmonellosis and vomiting may exhibit normal acidity. Vomiting causes metabolic alkalosis, which raises the pH, and salmonellosis causes moderate metabolic acidosis, which lowers pH. Together, the two effects on pH cancel each other leading to an almost normal pH in the presence of other symptoms. Diagnostic

systems for reasoning about compensating systems must reason about the symptoms of a group of causes, rather than reasoning serially about the symptoms for individual causes.

Synergistic symptoms have also been noted in medical practice. The synergistic symptom phenomenon occurs when contributions are additive and cross a threshold. For example, a patient with mild heart disease and lung infection is likely to have dispnea, a shortness of breath. Neither of the two causes is usually sufficient by itself to cause the effect, but taken together they go over a threshold and cause an additional observable effect.

9.2.3 *Discriminating among Hypotheses*

With some exceptions, most diagnostic tasks are limited more by resources for gathering data than by resources for reasoning or computation. In this common situation, the key challenge is not efficient decision making or weighing of evidence, but rather efficient gathering of evidence. This focuses attention on alternative strategies for acquiring data to discriminate among hypotheses. Evidence available at the beginning provides only a partial description of the initial state of the diagnostic situation. While such "presenting signs and symptoms" can be suggestive of abnormal or pathological conditions, in most cases conclusive evidence concerning an underlying cause must be obtained by additional steps. Furthermore, it is seldom practical to routinely obtain all of the possible data. There are costs, delays, and other factors that constrain the use of data-gathering operations.

When is there enough evidence to establish a fault or disease in diagnosis? Roughly speaking, for most diagnostic programs the answer is "when all of the other plausible alternatives are ruled out." Diagnosis proceeds by elimination rather than by seeking more and more evidence for a favorite hypothesis.

One reason favoring this competitive elimination approach is the multiple symptom phenomenon cited previously. Faults and diseases can cause multiple observable symptoms. Observing more of them does not necessarily make a favored hypothesis more certain. Diagnostic problem solving is usually posed in terms of a competition between mutually exclusive hypotheses.

The basic issue in discriminating among hypotheses is determining what data to acquire next. In troubleshooting, this is usually formulated in terms of the selection of new probe points. Alternatively, it could involve creating new "input test vectors" that would drive the system into different observable states. It could be reformulated in terms of the selection of test programs to run to test a computer system. In medical domains, the probe points are analogous to biomedical laboratory tests, noninvasive scanning techniques, or exploratory surgery. The test vectors are analogous to medical interventions, such as administering a drug. Most research in diagnosis systems has focused on probe selection. Although knowledge systems for generating test vectors have been built (Shirley, 1989), discussion of that topic goes beyond the scope of this chapter.

The most straightforward and widely used technique for selecting probe points is called the **guided-probe approach**. This approach starts at a point where a discrepancy has been detected. The idea is to follow it upstream to components that influence the output of the one where the discrepancy was noted. The search process focuses on those elements whose outputs are also discrepancies. The search continues until a component has been identified as having an incorrect output but whose inputs are correct. If the component receives valid information but produces a bad result, it must be faulted.