DIAGNOSTIC TOLERANCE FOR MISSING SENSOR DATA

Ethan A. Scarl

Boeing Computer Services
P.O.Box 24346 Seattle, WA 98124-0346

Abstract

For practical automated diagnostic systems to continue functioning after failure, they must not only be able to diagnose sensor failures but also be able to tolerate the absence of data from the faulty sensors. We show that conventional (associational) diagnostic methods will have combinatoric problems when trying to isolate faulty sensors, even if they adequately diagnose other components. Moreover, attempts to extend the operation of diagnostic capability past sensor failure will necessarily compound those difficulties. Model-based reasoning offers a structured alternative that has no special problems diagnosing faulty sensors and can operate gracefully when sensor data is missing.

Introduction

The success of space missions will depend critically upon the robustness of space systems' abilities to monitor, diagnose, and compensate for faults. Since sensors are at least as likely to fail as other components, an unforeseen loss of sensor information should degrade onboard diagnostic capabilities as little as possible. Ideally, diagnostic performance should degrade only to the extent that the information needed to isolate a fault is unavailable, and never because they are incapable of using the information which is still available in the altered environment.

Autonomous systems may be confronted by missing data for many reasons. For example, sensor polling machinery can fail, certain conditions may render a given sensor untrustworthy or invalid (e.g., the wrong fluid may be passing over a thermometer, or a radar may be tracking the wrong object), or a filter may reject the reading as spurious. The most common case, however, is the loss of data due to unpredicted sensor failure.

Knowing how to operate with missing data is distinct from the problem of recognizing the loss of a working sensor (diagnosis). It must be *known* that the data is missing. Both conventional methods and model-based reasoning may reach wrong conclusions if they try to make inferences from obsolete data instead of knowing that the data is not available. For some conditions, like a glitch in the sensor polling machinery, it is the responsibility of the supporting and interface systems to notify the diagnoser. On the other hand, the diagnoser will be the source of knowledge of a sensor failure, and may well have deduced the presence of an inappropriate sensor mode or environment.

Model-based reasoning permits a diagnostic algorithm that reasons explictly about the behavior of physical or computational system components and the connections between them. It will be seen that this algorithm approaches the ideal utilization of available information and does not suffer from combinatoric problems which afflict conventional methods trying to accommodate missing data from sensor failure.

The method suffers only from the requirement that system knowledge be properly represented and manipulated. Contrary to some common misconceptions, it is not necessary to have a model that is complete to any particular level of detail.

The next section describes the associational diagnostic approach most commonly used today, and which includes much of the work in rule-based expert systems as well as more traditional implementations of diagnostic logic with procedural languages. (Readers familiar with the shortcomings of conventional methods may wish to skip over this section.) The last section summarizes the model-based approach to diagnosis and then contrasts it with associational methods in its ability to diagnose sensor failure and to tolerate missing data.

Conventional Methods

Traditional approaches have tended to see diagnosis as the analysis of sensor data to determine the most likely faults. Automated diagnostics are usually implemented by rules or procedures which associate faults with patterns of sensed observation; hence the term *associational.*

Some of these inference methods chain forward directly from sensor data to faulty component likelihoods using an arbitrary distillation of expertise (rule- or table-driven associational methods), while others postulate faults, (manually) determine the resulting patterns of sensor readings, and finally reverse-index these faults by symptom (Failure Modes and Effects Analysis, or FMEA methods). In either case, the operating logic is to take sensor inputs and match them against patterns that indicate specific failures or classes of failure.

A matched pattern associates the present system condition with a predetermined failure. A successful match means that a fault has been located (or at least deemed likely), while an unsuccessful match means only that that particular fault mode is unlikely. Only after *all* fault modes have *failed* to match can the system be assumed to be operating properly. This is such an ungainly method for monitoring a healthy system that a number of applications have used model-based simulation (below) for monitoring, only to fall back on associational methods for fault location [e.g., Hofmann 88; Zwinglestein 85].

The implementation of an associational diagnostic inference may use production rules, a decision table, a programmed procedure, a decision tree, or an FMEA tree, but we will think of the knowledge it contains as equivalent to a collection of rules. A typical such rule might say that:

"if
  sensor A = 0, sensor B ε (100, 150), sensor C > 10, and sensor D = ON,
 then
  there an 80% chance that component E is broken and a 20% chance that component F is broken."

One might add:

"else
E and F are probably working, and there may even be nothing wrong with anything."

Broken Sensors

The first major problem is that this rule depends upon sensors A, B, C, and D all working properly. If any of them is faulty, the rule will deduce a wrong conclusion.

In principal, sensor failures would be inferred by similar rules, specially written for that purpose. These rules for determining sensor health are special in that they must be run *before* the above system health rule to prevent it from running with bad data; [Fox 83] refers to these as *meta-rules.*

As a simpler example, consider determining the health of four totally redundant sensors (call them P, Q, R, and S) which all measure the same parameter X. With perfect redundancy, it is possible to use simple "voting." A voting meta-rule might be:

"if P = Q, and Q = R, and P ≠ S, then S is faulty,"

Unfortunately, three more such rules are needed to determine the guilt of P, Q, and R. Note that four rules, one per sensor, are sufficient only because it is being assumed that no other sensors are relevant to X. Additional sensors would not only add more rules but would add complexity to the rules just given.

It is usually too burdensome to continually rerun all of these fault isolation rules just to be sure that everything is working. A speedup (if not a simplification) to determine that *nothing* is broken would be to use a special (hyper-meta- ?) rule for monitoring:

"if
P = Q and Q = R and R = S,
then
none of P, Q, R, or S is broken, so skip their fault isolation meta-rules and go directly to the rules that use P, Q, R, and S to diagnose other components."

Totally redundant sensors are not often available, and information from sensors that yield different but related quantities must be fused by less obvious meta-rules to infer the poor health of each in turn. For each sensor, another meta-rule must be written for each of the ways that its failure can be identified. This is rarely attempted for complex systems, and serious attempts have ended in failure due to the sheer number of cases that must be analysed, programmed, and validated [Delaune 85; Jamieson 86].

In summary, conventional associational methods have serious methodological and combinatoric difficulties in coping with sensor failure. This problem is often misleading to software developers, since they tend to feel that if each condition is coded methodically enough, or they just debug the most recent unexpected rule interactions, they will succeed. Unfortunately, the problem is inherent in

the associational approach. It is also worth noting that relatively minor subsequent changes made to the system may require substantial changes to the associational diagnostic code, with considerable effort expended in revalidation.

Missing Data

Missing sensor data corresponds to variables in the hypotheses of sensor-fault meta-rules being unbound. What happens to the inference that

"if P = Q, and Q = R, and P ≠ S, then S is faulty,"

after sensor P stops working? One possibility is that no rule using P is permitted to fire, so that an additional meta-rule like

"if Q = R, and Q ≠ S, then S is faulty,"

is needed for every possible combination of sensed data being present or absent, along with corresponding methods for propagating uncertainty, for every meta-rule. (Remember that the number of meta-rules is already likely to be a serious problem.) The only other alternative is to let the meta-rules fire in spite of missing data; they then must bear the responsibility of internally testing for all the possible combinations of missing data, and responding appropriately. In either case, special code must be written in advance for each such combination.

One may attribute associational reasoning's inadequate handling of breaking or broken sensors to its unprincipled mixing of structural and behavioral knowledge with environmental knowledge (sensor readings). This can force a great many repetitive representations of the same knowledge.

For example, consider again the four identical sensors P, Q, R, and S. Since the isolation of multiple faults involves combinatoric problems by any method (and since these combinatorics are more obviously expressed in the rules of associational systems than in the algorithms of model-based reasoning), let us consider the number of rules required for isolating a single fault and then for continued operation after multiple faults have occurred. This is not altogether unreasonable since multiple faults that did *not* happen simultaneously, and therefore can be diagnosed as single faults, will have a cumulative effect on continued operation. Our four sensors then require one monitoring rule, four diagnostic rules, 15 rules to cover all possible combinations of multiple breakage among sensors P, Q, R, and S in the monitoring rule, and 7 additional rules for *each* diagnostic rule (e.g., how to prove that S may be faulty if one or more of P, Q, and R are already known to be broken). This means that a total of 48 rules need to be written to support the continued diagnosis of this configuration after failures.

Of course, an actual associational architecture would try to minimize this by combining the repetitive representations in some way. We predict that such compaction will be achieved either through *ad hoc* mechanisms like deleting all conjunctive or disjunctive terms that mention a defunct sensor, or through references to something equivalent to a model-based representation.

The combinatorics of missing sensor data compound the combinatorics of sensor diagnosis. The traditional methods for developing and maintaining monitoring/diagnostic software are costly, time consuming, and incapable of handling situations that were unforeseen when the system was written. We conclude that associational methods are unsuitable for the complete and robust diagnosis of complex systems after failure.

Model-Based Generate and Test

An alternative approach has been (with differing nuances) variously referred to as "causal reasoning," "deep reasoning," "reasoning from first principles," and "reasoning from knowledge of structure and function." More recently, it has been simply called "model-based reasoning," because the "depth," "first principles," or "knowledge of structure and function" were invariably embedded in a model which (at some level of detail or abstraction) was isomorphic to the system, both in structure and behavior (function). This approach is described elsewhere [Davis 85, 88; Genesereth 85; Scarl 87, 88] in a variety of implementations, but will be summarized here.

The view embodied here is that all system components are initially under suspicion as possible causes of a perceived malfunction, and the job of *diagnosis* is to eliminate inconsistent suspects by showing that the assumption that they are responsible is contradicted by sensory observation. In model-based reasoning, the system is represented by some network of significant components or computational quantities whose outputs are determined by transfer functions upon their inputs. Suppose, for example, that components A and B have outputs connected to the inputs of component C, and that the output of C is directly measured by sensor S. When the states of A and B have been determined by setting their inputs, then their outputs determine the inputs to C, whose output in turn determines S.

The original conception was that these components corresponded one-to-one to physical system components, and that the structure of the model corresponded one-to-one to the connections between the physical components. This is often still true, but it has become clear that some systems need to have local properties (e.g., the voltage across a resistor) defined in terms of more global properties that are derived from the compaction or simultaneous analysis of different parts of the system (e.g., the current through a series circuit). This leads to the introduction of objects which represent global abstractions (e.g., total path resistance) rather than physical components.

An inversion facility is also required: given a component's output (by measurement, inference, or assumption), what can we say about its inputs? Usually (especially when a single point of failure is being sought), the inputs can be considered one at a time, with the others assumed to be as computed from the system commands. This uses a very broad sense of "inversion." If the component's transfer function is a "trapdoor function" that cannot be practically inverted, so that the output tells us nothing about the input, then the "inverse" is its whole domain. The trick is to be able to represent *whatever* information is thus provided.

The model is a behavioral simulation which can be used to monitor the system's operation. Choosing a set of commands (system inputs) for the model causes predictions to be computed for the system's sensors. Although the health of the system is again determined by matching sensor readings against these prescribed values, there are important differences from the associational approach:

- A match means a healthy system, instead of a fault

- The set of sensors to be matched can be computed dynamically and interactively

- The matching is done against values computed dynamically from system commands, rather than statically predetermined. (Note that associational rules could also compute the comparison value dynamically from commands, but, if so, they would be performing model-based reasoning rather than associational reasoning.)

Any discrepancy between predicted and measured values indicates either a failure in the physical system or an inaccuracy in the model. We will assume in this discussion that the model accurately describes the system's proper behavior.

Faults are located by generating hypotheses and testing those hypotheses against all available sensor data. Hypotheses usually are generated directly from sensor readings by using the inversion of the functional relationships in the model. A hypothesis typically concerns some particular object C, and has the form:

"The inputs of component C have the values expected for them, but C's output O is broken so that it has the unexpected value (or range of values, if analog) X."

For example, if the observed value of a discrepant sensor S is passed through inverted functionalities to component C, then the resulting hypothesis is a single fault in component C with the specific wrong value X at its output. C is then a *suspect* for the failure consistent with S. If C is the only suspect, then C has been determined to be the single point of failure, as determined by this model.

If a hypothesis is generated by simultaneously using all sensors, then it needs no further validation; since all available observational information has been used to manufacture it, there is none left that could contradict it. Usually, however, the hypothesis is generated from the reading of a single *discrepant* sensor that disagrees with the model's prediction, and all the other related sensors are used to verify the hypothesis. Whatever mechanism is used to generate a hypothesis, the hypothesis is testable in the model by inserting its hypothesized value (or range) for O in place of O's expected value.

Broken Sensors

Model-based reasoning has no need of any additional knowledge (such as meta-rules) to tell whether sensors are broken, but is able to simply treat a sensor like any other component [Scarl 88]. A sensor with a discrepant reading is *always* included as a suspect. A hypothesis will be generated to say that the sensor is broken so as to read what it actually did rather than what the model predicted for it. That hypothesis is tested for consistency with other sensor readings, just as any other component would be, and rejected or retained accordingly.

Thus, no special rules are needed just because a component is a sensor. Nor need sensors be cleared of suspicion before testing other objects. Instead, the hypotheses generated for *all* components are tested (simultaneously or in parallel) against current sensor readings, which, after all, constitute *all*

the useful knowledge available, and the *only* source of available knowledge (in addition to the model's knowledge of system structure and function). The inherent parallelism of the model-based approach is a distinct advantage: a faulty sensor hypothesis can be tested in parallel with hypotheses about other objects.

The complexity of diagnosing sensors is therefore no greater than diagnosing other types of component. In fact, certain assumptions can simplify the generate-and-test algorithm for sensors to the point of being trivial [Scarl 88]. This happens when the structure is such that sensors cannot be responsible for each other's discrepancies. If a maximum of N simultaneous faults is assumed, then more than N simultaneously discrepant sensors will validate each other. Thus, each sensor S that gives a discrepant reading is the *only* sensor in its associated suspect list (in the language of [de Kleer 87], its minimal conflict set). The existence of N other discrepant sensors will clear S of participation in an N-tuple failure.

This shortcut does not apply if sensed values are used for control (feedback), but it *does* apply to sensors whose control is independent of their actual readings. For example, an ammeter will control the current through its circuit by virtue of being in series, but it can be separated into a resistor and a virtual sensor. It is then its resistor and not its sensor that controls other sensors in the circuit, and any other discrepant sensor will clear the ammeter of being a single point of failure.

In summary, the model-based reasoning approach to finding faults in sensors is no more difficult than finding faults in other objects, and does not become more complex as more sensors are added to the system.

Missing Data

When data is lost, for any of the reasons mentioned in the Introduction, model-based reasoning takes account of that loss in rather obvious ways. First, we do not monitor the missing data, and so it cannot trigger diagnosis. Second, we simply exclude missing data from the pool of data against which failure hypotheses are tested. This is similar to deleting terms referring to faulty sensors from associational rules, but simpler and free of *ad hoc* character.

Any number and combination of sensor failures can therefore be handled in a straightforward and uniform way, with no need for special coding.

Returning to the four-sensor example from the discussion of associational diagnostics, if some fault is hypothesized which affects the predicted value of parameter X, then only those sensors known to be operational are used to confirm that hypothesis.

Recall that while only four diagnostic rules were required (and that only because no other system sensors were considered) by an associative diagnoser, 48 rules were required to continue diagnosis after sensor failures. Model-based reasoning, on the other hand, will perform all inferences required for continued operation using only four functional descriptions, one for each sensor, plus the information that they are all connected to X. Each functional description simply declares that its sensor's predicted reading is given by the value of its input parameter. In the KATE representation [Scarl 88], this functional description would simply be the name of (the output of) some object which is supposed to set parameter X. Furthermore, if all four sensors are identical, the four functional descriptions need not be written manually but may be inherited from a generic sensor type description.

The model-based method of handling missing data is conservative, in that fault hypotheses may be retained which could have been ruled out were the data available, but there is no possibility of wrongly abandoning a valid hypothesis.

In conclusion, a brief analysis has shown that traditional associative approaches to diagnosis cannot be extended effectively to operate after sensor failure, not only because structure and function is not explictly represented, but also due to the sheer numbers of redundant representations of the knowledge that is present. The model-based generate-and-test algorithm elegantly avoids these difficulties and promises much more robust diagnostic capabilities.

References

[Davis 85] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior," in D. G. Bobrow (ed.), Qualitative Reasoning about Physical Systems, MIT Press, Cambridge, MA, 1985, pp. 347-410.

[Davis 88] R. Davis and W. Hamscher, "Model-based Reasoning: Troubleshooting," in H. E. Shrobe (ed.), Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1988, pp. 297-346.

[de Kleer 87] J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, **32**, No. 1, April, 1987, pp. 97-130.

[Delaune 85] C. I. Delaune, E. A. Scarl, and J. R. Jamieson, "A Monitor and Diagnosis Program for the Shuttle Liquid Oxygen Loading Operation," *Proceedings of the First Annual Workshop on Robotics and Expert Systems*, Houston, TX, June, 1985.

[Fox 83] M. S. Fox, S. Lowenfeld, and P. Kleinosky, "Techniques for Sensor-Based Diagnosis," *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI-83)*, August, 1983, pp. 158-163.

[Genesereth 85] M. Genesereth, "The Use of Design Descriptions in Automated Diagnosis," in D. G. Bobrow (ed.), Qualitative Reasoning about Physical Systems, MIT Press, Cambridge, MA, 1985, pp. 411-436.

[Hofmann 88] A. G. Hofmann, J. G. Allard, L. B. Hawkinson, and M. Levin, "Object Oriented Simulation Models and the Uses in Real Time Expert Systems," *Proceedings of the Third Artificial Intelligence and Simulation Workshop at AAAI-88*, St. Paul, MN, August, 1988.

[Jamieson 86] J. R. Jamieson, "A Mandate for Autonomous Control and Monitor Systems (The Failure of Hard Automation)," 1986, *available from the author.*

[Scarl 87] E. A. Scarl, J. R. Jamieson, and C. I. Delaune, "Diagnosis and Sensor Validation through Knowledge of Structure and Function," *IEEE-Transactions on Systems, Man, and Cybernetics*, **SMC-17**, No. 3, May/June, 1987, pp. 360-368.

[Scarl 88] E. A. Scarl, J. R. Jamieson, and E. New, "Deriving Fault Location and Control from a Functional Model," *Proceedings of the Third IEEE Symposium on Intelligent Control*, Arlington, VA, August, 1988.

[Zwinglestein 85] G. Zwinglestein, J. L. Tyran, P. Bajard, "Failure Detection Based on Multilevel Dynamic Models Applied to a U-Tube Steam Generator for a PWR," *Proceedings of the Symposium on New Technology in Nuclear Power Plant*