Data Validation During Diagnosis, A Step Beyond Traditional Sensor Validation.

B. Chandrasekaran and W.F. Punch III[1]

Laboratory for Artificial Intelligence Research
The Ohio State University

Abstract

A well known problem in diagnosis is the difficulty of providing correct diagnostic conclusions in light incorrect or missing data. Traditional approaches to solving this problem, as typified in the domains of various complex mechanical systems, validate data by using various kinds of redundancy in sensor hardware. While such techniques are useful, we propose that another level of redundancy exists beyond the hardware level, the redundancy provided by expectations derived during diagnosis. That is, in the process of exploring the space of possible malfunctions, initial data and intermediate conclusions set up expectations of the characteristics of the final answer. These expectations then provide a basis for judging the validity of the derived answer. We will show how such expectation-based data validation is a natural part of diagnosis as performed by hierarchical classification expert systems.

1. Introduction

Diagnosis is the process of mapping system observations into zero or more possible malfunctions of the system's components. Most of the work in AI in diagnosis assumes the observations given to an expert system are reliable. However, in real-world situations, data is often unreliable and real-world diagnostic systems must be capable of taking this into account just as the human expert must. In this paper, we will discuss a knowledge-based approach to validation that relies on *diagnostic expectations* derived from the diagnostic process itself to identify possible unreliable data points.

Present-day aids for human experts performing diagnosis attempt to validate data before diagnosis begins. In the domains of various complex mechanical systems (Nuclear Power Plants, Chemical Manufacturing Plants, etc.), such aid is based on the concept of hardware redundancy of sensors. Each important system datum (pressure, temperature etc.) is monitored with a *number* of hardware sensors providing a redundancy of information from which a composite, more reliable value is extracted. Based on this hardware redundancy, a number of techniques were developed to validate a datum's value:

1. Providing multiple sensors of the same kind to monitor a datum. Loss of one sensor therefore does not preclude data gathering and any disagreements among the sensors can be resolved statistically. For example, to measure the temperature of a chemical reaction, multiple temperature sensors could be used in the reactor and their statistical average given as the overall temperature value.
2. Providing different kinds of sensors to monitor a datum. This situation provides the same redundancy as (1) as well as minimizing the possibility of some kinds of common fault problems. That is, certain events that inactivate one sensor type will not affect sensors of a different type. Continuing with the example of (1) above, half of the sensors might be thermocouples while the other half might be mechanical temperature sensors.
3. Using sensors in several different locations to infer a datum value. In this situation, data values are monitored both directly and inferred from other system data based on well-established relationships. For example, while the temperature of a closed vessel may be directly monitored, it can be inferred from the measurement of the pressure using the $PV = nrT$ equation.

Such hardware redundancy allows some data validation, but some limitations for this approach do exist:

1. The expense of installing and maintaining multiple sensors for each important datum greatly increases the cost of the mechanical system.
2. Common fault failures still happen, despite cautions mentioned above, especially as the result of severe operation malfunctions.
3. Human operators and engineers resolve many such diagnostic problems despite incorrect and even absent data. In other words, human experts are more tolerant of *bad* data whether it has been validated or not.

Therefore, while hardware redundancy does solve part of the problem, more sophisticated techniques are required to complete the job.

The following simple example will help in examining point (3) and other ideas[2]. Consider the mechanical system diagrammed in Figure 1 with data values indicated in Figure 2. It is a closed vessel with two subsystems, a cooling system and a pressure relief system. The vessel is a reactor which contains some process (nuclear fission, chemical reactions, etc.) that produces both heat and pressure. The data values of Figure 2 indicate that the temperature of the reactor vessel is above acceptable limits.

[1]We gratefully acknowledge the support of grants from the Air Force Office of Scientific Research, AFOSR-82-0255, and the National Science Foundation, CPE-8400840

[2]Note that the ideas presented here have been used on more complicated real-world systems [5, 7]. This example has been condensed from them for expository clarity.

Assume for the example that two possible causes of the high reactor temperature exist: either the cooling system has failed or the pressure relief system has failed and the added heat has overpowered the functioning cooling system. Given the sensor readings, what would be the diagnostic conclusion? The data conflict is the *normal* pressure and cooling system readings and the *abnormal* pressure relief system readings. The failure of the pressure relief system is plausible, data indicates its failure and no other system failure, but such a failure expects the pressure to be high! The step to take is to assume that both the pressure relief system failed and the pressure sensor is incorrect.

The process shown above demonstrates data validation at a higher level than that of simple sensor hardware validation. In the example, the pressure system has failed despite the lack of a high pressure datum. However, there is other strong evidence[3] that the pressure system has indeed failed. The human reasoner *expects* the pressure datum to be high since the preponderance of other data indicate a malfunction. That is, the human reasoner in pursuing likely diagnostic conclusions discovers a plausible diagnostic conclusion that meets all but (in this case) one expectation. The important points to note are that:

1. A diagnostic conclusion can and should be made based on the preponderance of other evidence.
2. The datum value that does not meet expectation should be questioned and further investigation of its true value made.

Note that this process involves redundancy, not at the level of sensor hardware, but at the level of *diagnostic expectation*. This is a redundancy of information that allows questioning (and subsequent validation) of data based on multiple expectations of diagnostic conclusions. If a conclusion is likely, but not all of its expectations are met, then those now questionable values are investigated by more computationally expensive techniques.

Such expectations can be the result of one of a number of processes. Deep models can provide information on expected data patterns for any diagnostic conclusion. From this information, judgments on the reliability of any of the actual data values can be made. Information provided from such deep models can be incorporated into compiled structures that can also provide information on data reliability. Finally, the expert himself can provide the information on data reliability to the diagnosis system based on his expert judgment of the particular diagnostic process, in effect acting as the deep model for the system.

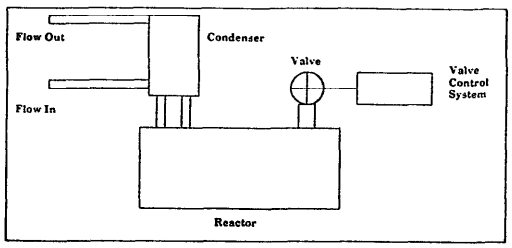

Figure 1: An Example Mechanical System

In this paper we will discuss compiled diagnostic systems that deal with conflicting datum at the level of diagnostic expectation indicated above. These are diagnostic systems that make conclusions based on diagnostic knowledge and some judgment on the validity of the data provided. In particular, we will show how

[3]In the example, this evidence is that there is a failure of the relief valve system which is part of the pressure system.

Variable	*Status*
Temperature	High
Pressure	Normal
Condenser	All sensors Normal
Cooling Water Flow System	All Sensors Normal
Relief Valve	Sensors indicate Malfunction
Valve Control System	All sensors Normal

Figure 2: Sensor Values for the Example

redundancy of diagnostic expectation is a natural extension to the hierarchical classification diagnostic model.

2. Diagnosis as a Hierarchical Classification Task

Significant interest has recently been directed towards understanding problem solving behaviors (diagnosis, planning, design) from the viewpoint of Information Processing Strategies. For example, Clancey [4] has shown that MYCIN is a species of classification problem solving activity. Earlier, in our work on MDX [2], we explicitly identified *hierarchical classification* as a strategy useful for some classes of diagnostic problem solving.

Diagnosis as a classification problem solving task is a matching of the data of the problem against a set of *malfunctions* (i.e., diseases, system failures, etc). If the present data are classified as a known malfunction, then the diagnosis is completed. Note that this is a *compiled* approach to diagnosis since it requires that the possible malfunctions and knowledge about how to establish those malfunctions be pre-enumerated. Other less well defined problems require a deeper model that rely on first principles (physics, chemistry, etc.) and an intimate understanding of the system at hand[4]. The rest of this section discusses the basic ideas behind hierarchical classification (See [6, 2] for details).

The malfunctions (diseases, failures) possible in the system are organized hierarchically. Typically, this hierarchy reflects a system:sub-system or function:sub-function relationship between the malfunctions[5]. Continuing with the example system in Figure 1, the malfunction hierarchy for the reactor system is shown in Figure 3. Each node in the malfunction hierarchy represents the hypothesis that some particular malfunction has occurred. Note that the nodes located in the upper levels of the hierarchy (Pressure System Failure, Cooling System Failure) represent more abstract malfunction hypotheses then those lower in the hierarchy (Relief Valve Failure, Condenser Failure). Further note that the sub nodes of any node are more particular kinds of the super node. For example, a Relief Value Failure is a particular kind of Pressure System Failure. Therefore, as one traverses the hierarchy in a top down fashion, one examines more detailed hypotheses about what malfunction has occurred in the system.

[4]Space limits this paper to the compiled system issues, see reference [9] for a detailed discussion of the deep model issues and computational strategies.

[5]Though the simple examples of this paper use only a single hierarchy, other work [10] recognizes that multiple hierarchies may be required to properly represent all system malfunctions.

Each node in the hierarchy has knowledge about the conditions under which the malfunction hypothesis it represents is plausible. Each node of the malfunction hierarchy is therefore a small expert system that evaluates whether the malfunction hypothesis it represents is present given the data. While there are a number of ways this could be accomplished, conceptually what is required is pattern–matching based on data *features*. Each node contains a set of features that are compared against the data. The results of this comparison indicate the likelihood of that particular malfunction being present. The pattern matching structure of a node in the CSRL language [1] is called a *knowledge group*. The knowledge groups compare relevant features against the data and yield a symbolic likelihood.

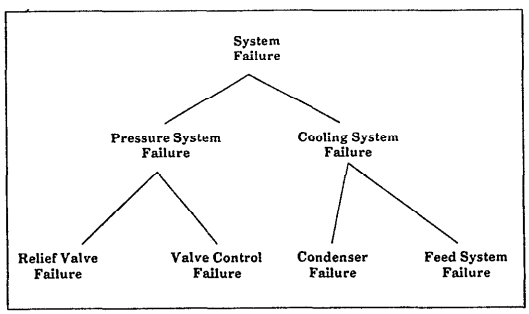

Figure 3: Hierarchy of malfunctions from Figure 1

Consider the knowledge group depicted in Figure 4 taken from the Cooling System Failure node of our example. The first section represents three queries about a datum value[6]. Each column of the table underneath represents a possible answer to each question (column 1 to question 1, etc.). The match value assigned to the knowledge group is based on the value located at the end of each row[7]. In our example when the answer to question 1 is True, the answer to question 2 is either High or Low and regardless of the answer to question 3, row 1 assigns a value of 3 to the knowledge group. The rows of the table are evaluated in order until either a row of queries matches or no row matches and a default value is assigned. Thus, when the data pattern of row 1 exists, the knowledge group (and thus the malfunction) is established at a high level of confidence.

Finally, the control strategy of a hierarchical classifier is termed *establish–refine*. In this strategy, each node is asked to establish how likely the malfunction hypothesis it represents is given the data. The node, using knowledge groups, determines an overall measure of likelihood. If the node establishes, i.e,. the malfunction is judged likely, then each sub of that node are asked to try and establish themselves. If a node is found to be unlikely, then that node is ruled–out and none its subs are evaluated.

Consider the example using the data from Figure 2 and the hierarchy of Figure 3. The top node is established since the temperature is high. Each of the subnodes is then asked to establish. Cooling System Failure rules out and none of its subnodes are examined. Pressure Relief Failure establishes and its subs are asked to try and establish themselves. This process

1) Is the Temperature Alarm on?
2) What is the Temperature above the Condenser
3) What is the Temperature of the Cooling Water Out?

1	2	3	Value
True	(OR High Low)	?	3
?	(OR High Low)	?	2
•••	•••	•••	•••

Figure 4: A Knowledge Group from Cooling System Failure

continues until there are no more nodes to examine. In this way, the most specific malfunctions that can be confirmed are given as the diagnostic conclusion[8].

3. Data Validation

Two important methods are available for validating data in conjunction with hierarchical classification. First, it is possible to establish a malfunction based on a preponderance of other evidence. If the node can establish but not all the data it expects is present, the data not meeting expectation is subject to question. In the original example, the Pressure Relief System Failure established despite a normal pressure reading based on a preponderance or other evidence.

Secondly, intermediate diagnostic conclusions from other nodes provide a context to evaluate data. If the Pressure System Failure does establish, its subs can expect the pressure reading to be abnormal. If it is not, they can also question the pressure reading.

In the remainder of this section, we will discuss the following aspects of a data validation system:

1. How data is questioned based on diagnostic expectations.
2. The various methodologies available that could resolve the questionable data.
3. How the normal control flow of diagnostic problem solving is affected.

3.1. Questioning a Datum Value

Discovering a questionable datum involves two steps. First, set some expectations, using local knowledge or the context of other nodes. Second, use those expectations to flag some particular data value as questionable.

The expectations of a malfunction are embodied in the knowledge group. The knowledge group mechanism was designed to give a rating of pattern fit to data. If the fit is not as expected, those data values not meeting expectations are identified as questionable. In the example of Pressure Relief Valve Failure, evidence exists that the valve has failed even though the pressure is normal. The lack of fit between data and pattern allow the pressure value to be identified as questionable. Diagnosis continues despite apparent data conflict since enough evidence exists for establishing the malfunction hypothesis.

[6]In the present CSRL implementation, these values are fetched from a database, though other means may be used, such as calls to deep models, simulations, etc.

[7]In this case, the values assigned are on a discrete scale from −3 to 3, −3 representing ruled–out and 3 representing confirmed.

[8]Note that if multiple conclusions are reached, then either multiple *independent* malfunctions have occurred or multiple *dependent* malfunctions must be resolved into a smaller set [8].

Furthermore, the expectations of previous nodes create a context of expectation for the node currently being examined. Consider the example hierarchy of Figure 3. In order to establish the malfunction hypothesis Relief Value Failure, the malfunction hypothesis Pressure System Failure must have established. In the context of considering the Valve Failure, some expectations were created based on the establishment of the Pressure System Failure node and other ancestors. Since these expectations always exist when considering Valve Failure, i.e., you can't get to Valve Failure without establishing Pressure System Failure, they can be coded into the Valve Failure Node.

How expectations are used for the Pressure Relief System Failure node is shown in Figure 5. A modification is made to the standard knowledge group of Figure 4 that allows the expert to indicate both a match value for the group and a set of data that do not meet the expectations established at this stage of the problem solving. Thus, Pressure Relief Failure establishes (based on other data features) despite the lack of a change of pressure. However, in establishing the node, one should question why the pressure did not change. This is done by placing the pressure value in the rightmost column of the matching row. If that row is matched, then the match value is returned but the indicated data value is placed in a list of questionable data values which will be examined later. If the match value is high enough, the node establishes despite the existence of conflicting data. That is, if there is enough evidence to show a malfunction despite a conflicting value, the problem solving may continue. However, it may be the case that the value being questioned is of vital importance to establishing the node. The match value will reflect this, the node will not establish, but the data will still be placed on the questionable data list. After an initial run of the problem solver, the questionable data list will consist of data values that did not meet the expectations of some node.

1) Is the Pressure Alarm on?
2) What is the Pressure above the Condenser?
3) Is the Temperature Alarm activated?

1	2	3	Value	Suspicious Data Value
False	(OR High Low)	True	1	Pressure
True	(OR High Low)	?	3	None
•••	•••	•••	•••	•••

Figure 5: Knowledge Group Modified for Data Validation

3.2. Questions Regarding Combinatorial Explosion

The knowledge engineer is responsible for providing the node with both feature matching data and datum values that do not meet expectations of the malfunction hypothesis at that point. This, as mentioned previously, is a compiled approach to data validation. It may appear that such a compilation of possibilities will result in a combinatorial explosion. However, not all possible combinations of sensor readings need be encoded; only those situations which the expert deems reasonable or necessary need be placed in the knowledge groups. More importantly, in our approach to use of knowledge groups, a hierarchy of abstractions is used to go from the data to the classificatory conclusion [3]. Thus, the set of data elements needed for any node in the hierarchy is limited to only those relevant for the malfunction hypothesis it represents. Furthermore, the data elements of each node are subdivided among the knowledge groups that need them. That is, even within the node, the data is further partitioned to only those knowledge groups that will use that data. The knowledge engineer is therefore presented with a much simpler task. Only those combinations of a few data items that present a missed expectation need be encoded into the diagnostic system

3.3. Resolving the Values of Questionable Data

A number of methods are available for resolution of questionable data. The system may examine the state of the nodes in the diagnostic hierarchy to see if any other nodes were satisfied or unsatisfied with the datum value in question. If many other nodes also questioned the same datum, increased evidence exists that the value is incorrect. In contrast, if no other node who used it questioned the value, evidence for its incorrectness is decreased and the validity of the knowledge group that originally questioned the value is suspect. This is a *redundancy of usage* by the nodes in the diagnostic hierarchy.

If a node indicates a data conflict that prevents it from establishing, that node can examine its subnodes to see if they have any expectations that resolve the conflict. Using Figure 3 as an example, assume that Cooling System Failure cannot establish because it requires that a temperature datum be high. The system could examine the subnodes of Cooling System Failure, namely Feed System Failure or Condenser Failure, to see if they can establish independent of the temperature reading. If Feed System Failure has enough data evidence to establish anyway, then its expectations might resolve the issue of the temperature reading. If it too expected a high temperature that does not exist, there is increased evidence that the temperature reading is incorrect. If it did not have any expectations about the temperature reading, then the knowledge found in the Cooling System Failure could be questionable. This is a *redundancy of expectation.*

The diagnostic system could contain more involved or detailed hardware checks then would be feasible to perform on all sensors as the data becomes available. General methods are invoked for a particular kind of sensor (thermocouples vs. mechanical thermal sensors) that are too computationally intensive/expensive to run all the time on incoming data. These procedures are run if there is evidence that their use would pay off. Such evidence is provided by a data conflict discovered by the diagnostic system. While these methods do not depend directly on information provided by the problem solving state (as the above methods do), they are only used when indicated by the diagnostic system. This is a *hardware redundancy of need.*

3.4. Control Issues

While sections 3.1 and 3.3 have discussed discovering and resolving possibly invalid data, this section addresses the issues of control flow changes to the normal establish – refine strategy.

1. What happens to data whose values have been proven to be either incorrect or unresolved? If found to be incorrect, the value in the data base must be modified, i.e., the central data base value modified, to indicate the change. If unresolved, it must be flagged as such in hopes of being resolved later.

2. If incorrect data has been found, then it is possible that the problem solver made some mistakes in its diagnosis. It may be necessary to re – run the hierarchy to see if the diagnostic conclusions change. Furthermore, any unresolved data may be resolved as a result of the new information.

3. The basic control strategy would then look like the following cycle:

a. Run the Hierarchy, finding questionable values.

b. Run resolution techniques of section 3.3 to resolve the values if possible.

c. Update the data base with any changes.

This cycle continues until either no data is questioned or the user sees an answer that is satisfactory.

4. Other control strategies are also available. The resolution techniques of section 3.3 could be run as soon as a data item is questioned, i.e., right in the middle of the problem solving. This requires a backtracking scheme that re-runs any other node that used that value. Finally, the operator can be directly involved in changing data values at any step of the process based on his/her expert opinion of the situation.

4. Implementation to Date

The majority of the structures and strategies indicated in the paper have been included into a version of the CSRL [1] tool as it has been applied to diagnosis of Cooling System Accidents in a Boiling Water Nuclear Power Plant. The knowledge groups have been modified as indicated in Figure 5 such that a datum value is questioned whenever it does not meet expectation. A number of scenarios of data conflict have been encoded into this system concerning the plant and more are being added. The control strategies that allow the hierarchy to be re-run until either no data is questioned or the user is satisfied have been implemented. Future work will concentrate on problems of backtracking to prevent re-running the entire hierarchy. With regards to data value resolution, present work is focusing on methodologies for resolving questionable sensors as indicated in Section 3.3. To date, each kind of sensor has associated with it a set of hardware checks that are not normally invoked. Furthermore, if the additional hardware checks do not resolve the problem, the user is presented with the conflict and information on all the nodes that used that value, including whether that value was questioned there or not.

5. Conclusion

While some data validation can be done using the standard techniques of hardware redundancy, a higher level of redundancy based on *diagnostic expectations* can address more of the issues of conflicting data in a more sensible manner. Intermediate diagnostic conclusions from hierarchical diagnosticians provide expectations that can indicate invalid data values. Note that the method is deceptively simple. This is due to the *focus* provided by chunking the match and expectation knowledge into the smaller, more manageable parts in the knowledge groups. Furthermore, the hierarchy also acts to simplify the complexity of the match and expectation knowledge due to the context of information provided by the establishment or rejection of parent nodes. Despite this, much power can be gained in both diagnosis and sensor/data validation by use of these simple methods. Some advantages include:

1. Questioning data based on diagnostic expectations provides a way to focus on only some data, as opposed to hardware methods which must worry about all data *a priori*.

2. Even if a data value is questioned, the diagnostic process can continue if other evidence exists for the failure. Thus the system can work with conflicting data.

3. Such a system integrates well with existing systems that presently rely solely on hardware redundancy by using those values as data for both diagnosis and a higher level of data validation.

4. The programming by a user of such a system is facilitated by existing tools (CSRL) that need only minor modifications.

Acknowledgments

We would like to thank Don Miller, Brian Hajek and Sia Hashemi of the Division of Nuclear Engineering at the Ohio State University for their help in developing and implementing these ideas. Also, one of the authors (Punch) would like to thank Mike Tanner for his invaluable aid in criticizing this paper and sharpening its ideas.

References

1. T. C. Bylander /S. Mittal. CSRL: A Language for Classificatory Problem Solving and Uncertainty Handling. *AI Magazine 7*(Summer 1986).

2. B. Chandrasekaran / S. Mittal. Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems. In *Advances in Computers*, M. Yovits, Ed., Academic Press, 1983, pp. 217–293.

3. B. Chandrasekaran. From Numbers to Symbols to Knowledge Structures: Pattern Recognition and Artificial Intelligence Perspectives on the Classification Task. In *Pattern Recognition in Practice–II*, North Holland Publishing, 1986, pp. 547–559.

4. W. J. Clancey. Heuristic Classification. *Artificial Intelligence 27*, 3 (1985), 289–350.

5. J. F. Davis / W. F. Punch III / S. K. Shum / B. Chandrasekaran. Application of Knowledge–Base Systems for the Diagnosis of Operating Problems. Presented to AIChe. Annual Meeting, Chicago Ill. 1985.

6. Gomez, F. / Chandrasekaran, B. Knowledge Organization and Distribution for Medical Diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics SMC–11*, 1 (January 1981), 34–42.

7. S. Hashemi, B.K. Hajek, D.W. Miller, B. Chandrasekaran, and J.R. Josephson. Expert Systems Application to Plant Diagnosis and Sensor Data Validation. *Proc. of the Sixth Power Plant Dynamics, Control and Testing Symposium*, Knoxville, Tennessee, April, 1986.

8. J. R. Josephson, B. Chandrasekaran and J. W. Smith, M.D. Abduction by Classification, Assembly, and Criticism. Revision of an earlier version called ``Abduction by Classification and Assembly" which appears in *Proc. Philosophy of Science Association, Vol.1, Biennial Meeting, 1986.* 1986.

9. V. Sembugamoorthy / B. Chandrasekaran. Functional representation of devices and compilation of diagnostic problem solving systems. In *Experience, Memory and Reasoning*, J. L. Kolodner and C. K. Riesbeck, Eds., Erlbaum, 1986, pp. 47–73.

10. J. Sticklen. MDX2: An Integrated Diagnostic Approach. Dissertation In Progress, Ohio State University. 1987.