Techniques for Sensor-Based Diagnosis

Mark S. Fox, Simon Lowenfeld*
and Pamela Kleinosky*

CMU-RI-TR-83-7

Intelligent Systems Laboratory
The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

*Westinghouse Electric Corporation
Pittsburgh, Pennsylvania

May 1983

This research was supported by the Westinghouse Research and Development Center.

Abstract

This paper describes a system called PDS, a forward chaining, rule-based architecture designed for the online, realtime diagnosis of machine processes. Two issues arise in the application of expert systems to the analysis of sensor-based data: spurious readings and sensor degradation. PDS implements techniques called retrospective analysis and meta-diagnosis as solutions to these problems. These techniques and our experiences in knowledge acquisition in a large organization, and the implementation of PDS as a portable diagnostic tool are described.

1. Introduction

Research in the field of AI diagnosis systems has been evolving rapidly since the first event-based (Nelson, 1982) or surface (Hart, 1982) reasoning systems (Shortliffe, 1976; Pople, 1977; Fox & Mostow, 1977; Duda et al., 1978), to systems that have functional or deep knowledge of their domain (Davis et al., 1982; Genesereth, 1982; Underwood, 1982; McDermott & Brooks, 1982). Whatever the style of diagnosis, these systems assume that information is provided manually through a question asking/answering dialogue, or automatically by means of sensors, or other devices. In both cases, the information is handled in the same manner, which, we have found, should not always be the case. In applications where the sources of information may be errorful (e.g., sensors), we found that it is just as important for a diagnostic system to reason *about* the sources of its information and their veracity, as it is to perform diagnosis based on the information.

During the summer of 1981, we began the design and construction of a rule-based architecture, called PDS, for the on-line, realtime diagnosis of malfunctions in machine processes. Diagnoses would be based on information acquired from tens to hundreds of sensors attached to a process. During the application of PDS, a number of sensor related problems arose. First, the process sensors in our applications degrade over time, reducing their diagnostic value. Second, a properly operating sensor may provide spurious readings periodically due to factors exogenous to the process. Though the frequency of such malfunctions are small, their detection may result in substantial savings. For example, in the electrical power utilities, replacement costs of electricity lost due to sensor malfunction averages $500,00/year per plant (Meiger et al., 1981). Any diagnosis system which is to receive its information directly from devices which possess these characteristics must be able to handle the information without providing incorrect diagnoses, or at least have its diagnosis degrade gracefully with the sensors. As a result, PDS was extended to deal gracefully with these problems. These extensions are the topics of this paper.

In the following, the architecture of PDS is described. We then examine the facilities provided by PDS to deal with sensor problems. Following, we discuss our experiences in the acquisition and testing of knowledge in PDS, and describe its implementation.

2. PDS: Basic Representation

PDS is a forward chaining rule-based system implemented in the Schema Representation Language SRL (Fox, 1979; Wright & Fox, 1982). The representation and propagation of belief is similar to that found in MYCIN (Shortliffe, 1976). For each rule, there are schemata describing each constituent part of the rule's antecedent (or evidence), a schema describing the rule's consequent (or hypothesis), and a schema describing the relationship between the rule's evidence and hypothesis. The implementation of rules as schemata results in an inference net similar to that found in PROSPECTOR (Duda et al., 1978). Sensor readings, hypotheses, and malfunctions are represented as sub-types of the **pds-node** schema (figure 2-1).

{{ **pds-node**

MB: "level of belief in the node being true"

MD: "level of disbelief in the node being true"

CF: "level of certainty = mb - md"

SUPPORTING-RULES: "rules for which this node is hypothesis"

SUPPORTED-RULES: "rules for which this node is evidence"

SIGNAL: "contains signal schema name(s)"

DESCRIPTION: "English description of the node"

HAS-IS-A: (or sensor hypothesis malfunction) }}

Figure 2-1: Generic Node in PDS

The following specializations (see HAS-IS-A slot) of **pds-nodes** are defined:

1. **sensor** schemata represent the actual sensors. Two additional slots are defined to store the current reading: READING-VALUE and READING-TIME.

2. **malfunction** schemata correspond to those states of the physical system which are indicative of the problem(s) to be diagnosed.

3. **hypothesis** schemata represent intermediate conclusions in the inference net.

A **belief-rule** schema (figure 2-2) represents those rules which are used by the inference program to propagate belief.

{{ **belief-rule**
EVIDENCE:
HYPOTHESIS:
SF-FUNCTION: "sufficiency function"
NF-FUNCTION: "necessity function"
SF: "value from SF-FUNCTION"
NF: "value from NF-FUNCTION"
CONTEXT: "context in which the rule fires"
DESCRIPTION: "English description of the rule" }}

Figure 2-2: **belief-rule** Schema

EVIDENCE is represented as a Boolean combination of nodes. The combination is explicitly represented by **and**, **or**, and **not** schemata. Each **belief-rule** has associated with it one or more contexts which are also represented as a Boolean combination. If the context is evaluated to be true, then the rule will be included in the diagnostic process.

Belief propagation is similar to the one used by MYCIN. To summarize, each hypothesis and malfunction has associated with it a measure of belief MB, and disbelief MD. These measures are altered by the evaluation of supporting **belief-rules**. In a **belief-rule**, the sufficiency of evidence is determined by evaluating the contents of the SF-FUNCTION slot. The necessity of evidence is determined by evaluating the contents of the NF-FUNCTION slot. Table 2-1 shows the direction of change of beliefs and disbeliefs as determined by evidence CF (the CF of disjunctive evidence (OR) is the maximum of the constituents, and the CF of conjunctive evidence is the weighted average of constituents (AND) or the mimimum of constituents (FAND)), and the rule's SF and NF (an up-arrow indicates an increase, a dash indicates no change). Actual updating is performed using MYCIN's updating rule (e.g., $MB_{new} = MB_{old} + (1 - MB_{old}) * \text{rule-support}$).

Rule: / Evidence:	Sufficiency SF > 0	Sufficiency SF < 0	Necessity NF > 0	Necessity NF < 0
Present (CF[E]>0)	↑ MB[H]	↑ MD[H]	---	---
Missing (CF[E]<0)	---	---	↑ MD[H]	↑ MB[H]

Table 2-1: Belief Revision Definition

A **pds-node** may have one or more **signals** attached to it (Figure 2-3). When the node with which a **signal** is associated has a certainty factor CF in the range specified in the signal definition, then the text in the MESSAGE slot is displayed and the function whose name is in the action slot is evaluated (this may trigger an alarm, for example).

{{ **signal**
MIN-RANGE: <constant>
MAX-RANGE: <constant>
MESSAGE: "text to be displayed"
ACTION: <function-name>
DESCRIPTION: "English text describing the signal" }}

Figure 2-3: signal Schema

A **context** (Figure 2-4) specifies an external condition which may be relevant to the execution of the diagnostics. For example, when diagnosing a machine, different rules may apply at start-up than found during normal operation. For each rule, a Boolean of contexts can be defined, such that the rule will fire only when the context specification is true.

{{ **context**
VALUE: <true | false | 0 | 1>
DESCRIPTION: "English description of this context" }}

Figure 2-4: context Schema

As found in most knowledge representation systems, SRL provides the user with the ability to define type hierarchies of schemata using the **is-a** relation, and instances of types using the **instance** relation. Rules may refer to types of hypotheses, sensors, and malfunctions, rather than instances. When the knowledge network for a particular machine is created, instances of sensors, etc. inherit rules attached to nodes in their related type hierarchy. A library of rules may be created for different types of machines, and instantiated with little effort.

3. Retrospective Analysis

Spurious readings do occur often enough in sensors to require their detection and omission from the diagnostic process. These readings may be due to factors exogenous to the process, or to sensor malfunction. Spurious readings are handled in most diagnostic systems before they reach the system: the readings are smoothed or omitted, manually or by a pre-processor associated with the sensor. It quickly became apparent in our applications that such an approach was not sufficient. First, external modification of readings prohibit the system from performing other types of analyses not anticipated in the design of the sensor and its pre-processor. Second, we found that retrospective analysis of the unaltered sensor data was important in order to refine the rule base.

Solutions to this problem were found to have much in common with other diagnostic techniques. In particular, a variety of time series analyses was found to be important. Rate of change (first derivative), averages, filtering, and curve smoothing are examples of the kinds of time domain analysis employed both at the front end of diagnostic systems, and during the diagnosis itself.

To provide general retrospective analysis support, PDS provides the ability to store and analyze successive readings of a sensor, or the successive values of any other node. A **reading-set** schema (figure 3-1) acts as a memory for PDS. Successive readings/values are stored in the READING-LIST slot. Information about a reading, e.g., time of reading, may be attached directly to it using SRL's facility for attaching meta-schemata to a schema, slot, and/or value.

{{ **reading-set**
READING-LIST: "list of readings"
LIST-SIZE: "maximum number of readings in the list"
FUNCTION: "name of function which analyzes the reading list"
STATISTIC: "result of applying the function to the data"
MB: "belief in STATISTIC"
MD: "disbelief in STATISTIC"
SUPPORTED-RULES: "rules for which this node is evidence"
SUPPORTING-RULES: "rules for which this node is hypothesis"
DESCRIPTION: "English description of this node" }}

Figure 3-1: *reading-set* Schema

Another type of rule, called **reading-transform** (figure 3-2), is the link between some input node, usually, but not necessarily a **sensor** node, and a **reading-set** node. When a **reading-transform** rule fires, its TRANSFORM function is applied to the value found in the rule's EVIDENCE node. If the evidence node is a **sensor**, the value is a **sensor** READING, otherwise it is the node's certainty factor. The transform is useful for such things as conversions of engineering units, scaling, etc. The result of the transformation is automatically appended to the READING-LIST of the **reading-set** node. When all

the rules leading to a **reading-set** node have fired, the FUNCTION specified in the reading-set schema is applied to the READING-LIST. This function does all the required signal processing and the result is placed in the STATISTIC slot of the **reading-set** node. The STATISTIC can, from this point on, be used as a "normal" sensor reading.

{{ **reading-transform**
INPUT-NODE: "Name of node supplying input to transform"
READING-SET-NODE: "Destination of result of transform"
TRANSFORM: "Name of transform function"
DESCRIPTION: "English description of this node" }}

Figure 3-2: reading-transform Schema

---

4. Meta-Diagnosis

The sensor intensive applications of PDS share the problem of sensor degradation; environments containing corrosive chemicals and widely varying temperatures can reduce sensor performance. This problem has been solved partially at the machine level by the placement of redundant, overlapping sensors. But at the diagnosis level, the problem of recognizing and removing malfunctioning sensors from the diagnostic process has received little attention in the AI literature.

If the rules which used sensors as evidence only referred to a single sensor, then it might be possible to summarize the acceptability of a sensor's reading by the CF associated with it. Setting the CF to zero would stop propagation of any belief or disbelief by the supported rule. (The change in belief provided by a rule is the product of either the rule's SF or NF and the evidence's CF.) But in many applications sensors may be combined as evidence in a single rule. Using CF's to reflect sensor degradation in such systems would have an unexpected result. Consider the following:

1. If evidence is conjunctive and the fuzzy minimum operator (FAND) is used to derive the evidential CF, then the existence of a zero CF would remove the entire rule though the other sensors may be operating and overlap the redundant sensor.

2. If evidence is conjunctive and a weighted average is used to derive the evidential CF, then the sensor(s) with a zero CF would reduce the rule's change in belief since its weight is not reduced (PDS uses a weighted average to derive the evidential CF).

Neither approach solves the problem satisfactorily.

We call the solution implemented in PDS *meta-diagnosis*. The first step in meta-diagnosis is the detection of sensor degradation. This is accomplished through the use of rules which monitor a

sensor's behavior. The ultimate consequent of these rules is one or more sensor **malfunction** schemata. The second step is the adaptation of rules to reflect the reduction in importance of a malfunctioning sensor in the diagnostic process. This is accomplished through the introduction of a **parametric-alteration** rule (figure 4-1). This rule provides the capability of altering the definition of any other node or rule in the system. A **parametric-alteration** rule may be attached to any node. When the EVIDENCE-SLOT in the node changes, the rule's TRANSFORM function is applied to the value, possibly altering the HYPOTHESIS schema.

{{ **parametric-alteration**
IS-A: rule
EVIDENCE: "the schema monitored by this rule"
EVIDENCE-SLOT: "the slot whose value is monitored by this rule"
TRANSFORM: "function whose result is placed
in the hypothesis slot"
HYPOTHESIS: "schema altered by this rule"
HYPOTHESIS-SLOT: "slot in hypothesis altered by this rule" }}

Figure 4-1: parametric-alteration Schema

An example of the use of this type of rule is the reduction of a sensor's weight in the conjunctive evidence of a belief-rule. It is important to note that this is a meta-rule in the sense that it alters the rule base, while not performing actual diagnosis. Such rules must be used carefully as they may introduce cycles in the propagation of belief.

5. Composite Sensors

An alternative method of analyzing and reacting to readings from redundant overlapping sensors, which falls short of altering the rule base as provided by meta-diagnosis, is to combine multiple sensors into a single composite sensor. In many domains, techniques such as:

- voting, and
- auctioneering (i.e., ignoring the lowest or highest reading)

are used to combine sensor readings. PDS supports the exploration of these types of techniques by means of a **composite-sensor** schema (figure 5-1).

{{ **composite-sensor**
IS-A: pds-node
SENSOR: "list of sensors whose readings are to be combined"
TRANSFORM: "function which generates a composite reading"
READING-VALUE: }}

Figure 5-1: composite-sensor schema

The TRANSFORM slot contains a lisp function which analyzes the individual readings of the sensors listed in the SENSOR slot, and fills the READING-VALUE, MB, and MD slots with the composite reading. Standard functions can be provided to implement voting, auctioneering, and other multiple sensor techniques.

6. Knowledge Acquisition and Testing

A number of systems capable of performing on-line diagnosis in an industrial environment are under development at the Westinghouse Electric Corporation. A working prototype of one of these systems is being installed and tested at this time and is the subject of the following discussion. Although many of the steps in the development of this prototype are common ones to the creator of an expert system, they often brought unexpected results when carried out in the commercial environment of a single large company.

The knowledge engineer responsible for the generation of rules for this system was actually a team. The person writing the rules was an engineer whose background was in the problem being diagnosed (a quasi-expert) and whose acquaintance with PDS consisted of a working knowledge of the general principles and those structures which applied specifically to the problem at hand. Working in close conjunction with this "quasi-expert" was an engineer whose background was in expert systems and who knew PDS in depth.

As with the development of most systems of this type, the first step was the creation of a small system that we called a "root system". It began with ten sensors and used forty-four rules and twenty-nine intermediate hypotheses to indicate seven malfunctions. It took approximately one month to develop and test. This "root" served us in two ways. First, it acted as a vehicle for eliciting information and stimulating the thinking of the experts. Secondly, it was a tool to sell the experts on the feasibility of what they were being asked to undertake and, more importantly, to sell upper management on the probability of a return on what they were being asked to invest.

The testing scheme developed for the "root system" was a general one, and with some expansion was used for the prototype itself. First, approximately 150 sets of test data were generated in four groups. Group one tested the interactions of the rules themselves. Group two tested the system's response to malfunctions previously diagnosed by the experts. Group three tested the experts'

response to malfunctions diagnosed by the system. Group four comparison-tested the responses of the experts and the system to data neither had "seen" before.

The second phase of the testing is currently under way, and involves the installation and operation of the sensors used in the diagnosis system in an industrial environment. The resulting data are fed directly to the diagnosis system and the presence of any malfunctions are noted, recorded and displayed.

Preliminary results of this testing are promising. Although mathematical analysis of these results has not yet been performed, the expert analyses of the data have agreed quite closely with the computer analyses.

No development project is without its problems and this one has been no exception. Our first major problem was in convincing the necessary experts to participate in the project. Older and more established engineers, most useful as experts, have been traditionally wary of computer systems and the perceived possibilities of being replaced by a machine. Also the necessary experts were distributed throughout the company, raising many organizational problems.

Once a sufficient number of experts agreed to participate in the project, we discovered the second major difficulty; problems within the cognitive models of the experts themselves. When the experts were questioned about the if-then rules used in their thinking processes their first tendency was to go from a sensor reading to a final malfunction in one step. This is not to say that one-step diagnoses are not possible, just uncommon. After some work, the experts began to think of their diagnosis rules step-by-step, but then the difficulties of verbalizing the steps became evident; at this point the quasi-expert status of the knowledge engineer became very valuable. After the rules had been established to the satisfaction of the experts and the knowledge engineer, quantifying the steps in terms of sufficiency and necessity functions became the final hurdle. For this step of the process a script was developed using the form of the questions that, through trial and error, seemed to elicit the most consistent responses from the experts. This script was then used for all experts evaluating all rules. The fact that most engineers are very logic-oriented and generally resent what they consider to be a need to justify their decision-making processes should not be discounted at any step of the cognitive process.

The third problem encountered was an actual gap in the knowledge of the experts. The expert diagnosis of the system under consideration is in itself a new problem and the knowledge of sensor malfunction and the forms it may take is extremely limited. At this time the problem is being addressed by using the data from the on-line test as a basis for the on-going development of rules relating to sensor behavior.

The sensors were also the source of another problem, this time in the testing of the system. Since the sensors being used for the on-line test malfunctioned more often than desired, it was difficult to obtain data to test anything but malfunction detection rules. It is expected that this problem can be overcome by the use of more efficient sensor systems currently being designed.

The most difficult problem discovered during development was the disagreement of the experts on

certain rules of diagnosis. It is difficult for a knowledge engineer to offer any solution to this problem. When the problem occurred an attempt was made to hold group meetings of the experts to discuss the disagreements and reach a consensus. If no agreement could be reached, the rule was modified, usually making it less effective. Fortunately, this occurred infrequently and did not unduly reduce the efficacy of the system.

Some "rules of thumb" became obvious during the project and when followed gave optimum results for the time involved. It is probably best to present then in list form:

1. Do not misrepresent the capabilities of the system. It will alienate the experts and raise management expectations to a level the system cannot deliver.

2. Develop a script for use in questioning the experts. If the same question is asked two ways, it will often elicit answers as if two different questions had been asked.

3. After a phase of development is completed (by judgement of the knowledge engineer) turn the system over to any interested experts. This often results in information that was missed during questioning.

4. For best efficiency, the team approach to knowledge engineering seems to be successful. The quasi-expert knew when steps were left out and could offer the experts assistance in verbalizing their thought processes, while the PDS expert insured that the most efficient PDS structures were used throughout.

7. Implementation

As mentioned earlier, PDS is written in SRL, which in turn is implemented in the Franz Lisp dialect of LISP running on a VAX-780, under the VMS operating system. The program consist of four parts: the knowledge-base development functions, the input simulation functions, the inference mechanism and an explanation facility.

The knowledge-base development part of PDS provides the "tools" that the knowledge engineer, or the sophisticated expert, can use to develop the rule base. Aside from the obvious functions that ease the addition, deletion and editing of rules, an extensive library of utility functions exists to list and describe the rules, to save and restore the rule base, to print hard copy listings, to initialize the system to a predefined state, and so on.

Testing of the rules is facilitated by the presence of functions that allow the manual entry of sensor values and the setting of specific contexts. Input data can be entered in lieu of actual sensor readings. An edit function is provided to allow the study of the effects of small modifications in sensor values on the propagation of belief process.

The inference program performs forward propagation of belief from sensor nodes. If a parametric alteration rule fires, all nodes and rules directly or indirectly affected are re-evaluated.

The explanation facility is quite primitive. It does not involve natural language generation, but rather it puts together sentences from "canned" fragments contained in the description slot of the various schemata. Plans exist to improve on this particular feature, both in the language generation aspect and in the range of questions it can answer (the only question at this time is "why ?"). The explanation facility can be connected to a text-to-speech converter (the PROSE 2000 board by Telesensory,Inc.), thereby being able to "speak" its explanations.

A goal of this project is to provide machine technicians with a portable and inexpensive diagnostic tool. A microprocessor version of PDS is being implemented on a four board package (CPU, memory, graphics I/O and text-to-speech converter), that fits in a hand-carry suitcase. We call it the "Expert in a Box" version of PDS. It has two modes of operation. In the independent mode, it can perform diagnosis only. In the alternative mode, it can connect, via a built-in modem, to the VAX, and thereby act as an intelligent remote terminal for the expert system. In this mode, it can also be connected to a loudspeaker, for voice explanations, and to a color graphic display.

8. Conclusions

PDS is a forward chaining rule-based system for process diagnosis. It is being implemented in environments in which data acquisition is totally automated, thereby limiting the amount of user interaction. In pursuing its implementation in these environments, problems of spurious readings and general degradation had to be solved. We chose to solve these problems in three ways. First, raw data was introduced, stored, and retrospectively analyzed in the knowledge base in order to provide greater flexibility. Second, meta-diagnosis was performed to adapt the rule-base to changes in its physical environment (i.e., sensor degradation); as sensors degrade, PDS focuses its analysis only on the sensors which provide reliable information. Third, redundant sensor readings were analyzed and combined into a composite sensor. The approach reported here is just a step towards the general problem of the intelligent acquisition and analysis of sensor-based information. Techniques such as sensor redirection and tuning remain to be investigated.

9. Acknowledgements

Chris Kemper, a domain expert and system user, provided valuable feedback on the design of PDS. R. Byford and A.I Szabo provided continued management support.

10. References

Davis R., H. Shrobe, W. Hanscher, K. Wieckert, M. Shirley, and S. Polit, (1982), "Diagnosis Based on Description of Structure and Function", *Proceedings of the American Association for Artificial Intelligence*, Aug. 1982, pp. 137-142.

Duda R.O., P.E. Hart, P. Barrett, J.G. Gaschnig, K. Konolige, R. Reboh, and J. Slocum, (1978), "Development of the Prospector Consultation System for Mineral Exploration: Final Report", Tech. Rep., SRI International, Menlo Park CA, Oct. 1978.

Fox M.S., (1979), "On Inheritance in Knowledge Representation", *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo Japan.

Fox M.S. and D.J. Mostow, (1977), "Maximal Consistent Interpretations of Errorful Data In Hierarchically Modeled Domains", *Fifth International Joint Conference on Artificial Intelligence*, Cambridge MA, 1977.

Genesereth M.R., (1982), "Diagnosis Using Hierarchical Design Models", *Proceedings of the Second Conference of the American Association for Artificial Intelligence*, Aug. 1982, pp. 278-283.

Hart P., (1982), "Direction for AI in the Eighties", *SIGART Newsletter*, No. 79, Jan. 1982.

McDermott D. and R. Brooks, (1982), "ARBY: Diagnosis with Shallow Causal Models", *Proceedings of the Second Conference of the American Association for Artificial Intelligence*, Pittsburgh PA.

Meijer C.H., J.P. Pasquenza, J.C. Deckert, J.L. Fisher, D.B. Laning, and A. Ray, (1981), "Online Power Plant Signal Validation Technique Utilizing Parity-Space Representation and Analytic Redundancy", Electric Power Research Institute, Technical Report EPRI NP-2110, Nov. 1981.

Nelson W.R., (1982), "REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents", *Proceedings of the Second Conference of the American Association for Artificial Intelligence*, Aug. 1982, pp. 296-301.

Pople H., (1977), The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Aug. 77.

Shortliffe E.H., (1976), *Computer-Based Medical Consultations: MYCIN*, New York: American Elsevier.

Underwood W.E., (1982), "A CSA Model-based Nuclear Power Plant Consultant", *Proceedings of the Second Conference of the American Association for Artificial Intelligence*, Aug. 1982, pp. 302-305.

Wright J.M., and Fox M.S., (1982), "SRL/1.5 User Manual", Robotics Institute, Carnegie-Mellon University, Pittsburgh PA.