Refining the Knowledge Base of a Diagnostic Expert System: An Application of Failure-Driven Learning

Michael J. Pazzani
The Aerospace Corporation
P.O.Box 92957
Los Angeles, CA 90009

Abstract

This paper discusses an application of failure-driven learning to the construction of the knowledge base of a diagnostic expert system. Diagnosis heuristics (i.e., efficient rules which encode empirical associations between atypical device behavior and device failures) are learned from information implicit in device models. This approach is desireable since less effort is required to obtain information about device functionality and connectivity to define device models than to encode and debug diagnosis heuristics from a domain expert.

We give results of applying this technique in an expert system for the diagnosis of failures in the attitude control system of the DSCS-III satellite. The system is fully implemented in a combination of LISP and PROLOG on a Symbolics 3600. The results indicate that realistic applications can be built using this approach. The performance of the diagnostic expert system after learning is equivalent to and, in some cases, better than the performace of the expert system with rules supplied by a domain expert.

Introduction

An important part of the construction of an expert system is the development of the knowledge base. This paper describes an application of computer learning to the construction of the knowledge base of an expert system for the diagnosis of anomalies in the attitude control system of a satellite. (The attitude control system is responsible for detecting and correcting deviations from the desired orientation of the satellite.) Rather than inducing diagnosis heuristics (i.e., empirical associations between symptoms and device failures) from a number of training examples, diagnosis heuristics are deduced as needed from device models.

The techniques illustrated in this paper are applicable to learning diagnosis heuristics for complex systems such as a power plant or a satellite. The status of such systems is continuously monitored for unusual or atypical features. When one or more atypical features are detected, a diagnosis process seeks to find an explanation for the atypical features. This explanation typically involves isolating the cause of the atypical feature to a component failure. Occasionly, the explanation may be that system is in a normal but unusual mode.

Two different approaches have been used for fault diagnosis. In one approach [3, 6], the observed functionality of devices are compared to their predicted functionality which is specified by a quantitative or qualitative model of the device [4]. For a large system, comparing observed to predicted functionality can be costly. The alternative approach [11, 14] encodes empirical associations between unusual behavior and faulty components as heuristic rules. This approach requires extensive debugging of the knowledge base to identify the precise conditions which indicate a particular fault is present.

We describe the the Attitude Control Expert System (ACES) which integrates model-based and heuristic-based diagnosis. Heuristics examine the atypical features and hypothesize potential faults. Device models confirm or deny hypothesized faults. Thus, heuristics focus diagnosis by determining which device in a large system might be at fault. Device models determine if that device is indeed responsible for the atypical features.

The initial diagnosis heuristics used in ACES are quite simple. They often hypothesize faults which are later denied by device models. We call this a **hypothesis failure**. When a fault is proposed, and later denied by device models, the reasons for this hypothesis failure are noted and the heuristic which suggested the fault is revised so that the hypothesis will not be proposed in future similar cases. This is a kind of failure-driven learning [12] which enables a diagnostic expert system to start with heuristics which indicate some of the signs (or symptoms) of a failure. As the expert system solves problems, the heuristics are revised to determine what part of the device model should be consulted to distinguish one fault from another fault with similar features. There are several reasons why this approach is desirable:

- Device models are a natural way of expressing the functionality of a component. However, they are not the most natural or efficient representation for diagnosis [13].
- Determining some of the signs of a fault (i.e., the initial diagnostic heuristics) is a relatively easy task. Often, the initial fault diagnosis heuristics are definitional. For example, ACES starts with a heuristic which states that if a tachometer is reading 0, then it is faulty. Later this heuristic is revised to include conditions to distinguish a fault in a tachometer from a fault in the component measured by the tachometer.

One way to view failure-driven learning is as an extension of dependency-directed backtracking [15]. In dependency-directed backtracking, when a hypothesis failure occurs, the search tree of the current problem is pruned by removing those states which would lead to failure for the same reason. In failure-driven learning, the reason for hypothesis failure is recorded, so that the search tree of future similar problems does not include states which would lead to failure for the same reason.

Failure-driven learning dictates two important facets of learning: when to learn (when a hypothesis failure occurs) and what to learn (features which distinguish a fault in one component from faults in other components). What is not specified is how to learn. For example, a learning system could learn to distinguish a faulty tachometer from failures with similar features by correlation over a number of examples (e.g. [7, 8, 16]). Device models (or a teacher) could classify a large number of examples as positive or negative examples of broken tachometers. For example, the heuristic which suggests broken tachometers could be revised to include a description of those combination of features which are present when a tachometer is faulty, but not present when the tachometer is working properly. In contrast, ACES learns how to avoid a hypothesis a failure after just one example. It does this by finding the most general reason for the hypothesis failure. The device models serve a dual role here. First, they identify when to learn by denying a hypothesis. More importantly, they provide an explanation for the hypothesis failure. The device models indicate which features would have been needed to be present (or absent) to confirm the hypothesis. This deductive approach to learning is called **explanation-based learning** [5, 9]. Explanation-based learning improves the performance of ACES by creating fault diagnosis heuristics from information implicit in the device models.

Schank [12] has proposed failure-driven learning as the mechanism by which a person's memory of events and generalized events evolves with experience. A person's memory provides expectations for understanding natural language understanding and inferring other's plans and goals. When a new event fails to conform to these expectations, it is stored in memory along with the explanation for the failure to prevent the generation of the erroneous expectation in the future. In future similar situations, this event will be the source of expectations rather than the generalized event whose

expectations were incorrect. In failure-driven learning as applied to fault diagnosis, the failures are of fault hypotheses as opposed to expectations. The reason for failure is identified as some aspect of a device's function which disagrees with the fault hypothesis. The correction is to modify the heuristic rule which proposed the incorrect hypothesis to check that aspect of the device before proposing the fault.

Failure-driven Learning of Diagnosis Heuristics

In this section, we describe our approach to learning fault diagnosis heuristics by finding symptoms of faults implicit in device models. First, let us clarify what we mean by a device model. Following Chandraskaran [13], we represent the following aspects of a device:

- **Structure:** Specifies the connectivity of a device.
- **Functionality:** Specifies the output of a device as a function of its inputs (and possibly state information).

It is not important to the expert system or the learning module that the functionality be expressed quantitatively or qualitatively. The important part is that given the observed inputs of a device, the device model can make a prediction about the output. The predicted value of the output be compared to the observed value or can be treated as an input to another device.

Reasons for Hypothesis Failure

We have identified three different reasons for failing to confirm a hypothesis. For each reason, we have implemented a correction strategy.

- **Hypothesized Fault Inconsistent Prediction:** The hypothesized failure is inconsistent with observed behavior of the system. The strategy for correction is to check for other features which the proposed fault might cause.
- **Hypothesized Unusual Mode– Enablement Violated:** The atypical features can be explained by the system being in a normal but unusual mode. However, the enabling conditions for that mode are not met. The strategy for correction is to consider an enabling condition of the unusual state.
- **Hypothesized Fault– Unusual Input:** The device hypothesized to be faulty is in fact functioning properly. This typically occurs when the input to a device is very unusual. In this case, the output of the device is unusual and the device might be assumed to be faulty unless the input is considered. The strategy for correction is to consider the device functionality.

Revising Fault Diagnosis Heuristics

When there is a hypothesis failure, the explanation for the failure is found and the heuristic rule which proposed the hypothesis is revised. A heuristic rule which proposes a fault can apply to one particular component (e.g., the light bulb of the left taillight) or a class of components (e.g., light bulbs). Similarly, the correction strategy can apply to a particular component or a class of components. The manner in which the knowledge base of heuristic rules is revised depends on the generality of the heuristic rule and correction. These interact in the following manner:

- **Heuristic rule not more general than the correction:** The correction is added to the heuristic rule and this new more specialized rule replaces the old rule.
- **Heuristic rule more general than the correction:** The correction is added to the heuristic rule and applied only in the specialized case. The old rule is retained for use in other cases.

There are two other issues to be considered in revising heuristic rules. First, since some testing is being added to hypothesis generation, it would be wasteful to repeat the same test during confirmation. To avoid this potential problem, the revision to a rule caches the results of a test. Second, the amount of search necessary to prove a conjunction of subgoals in PROLOG (the language we use to implement our rules) is dependent on the order in which the subgoals are attempted. We use a strategy to order the tests in a revised rule similar to one proposed by Naish [10]. This strategy minimizes the size of the search space by detecting the ultimate failure of a rule as soon as possible. This assumes that decreasing the search space is the best means of increasing performance. This is true in our application since testing for the presence or absence of any feature is equally expensive. Cantone [1] gives an approach for ordering tests based in part on the cost of the test.

A Definition of Failure-driven Learning of Fault Diagnosis Heuristics

More formally, a diagnosis heuristic can viewed as the implication:

F and consistent(H) → H

where F is a set of features, H is a hypothesis, and **consistent(H)** is true if believing H does not result in a contradiction. (See [2], for a discussion of **consistent**.) In our approach to learning and fault diagnosis, **consistent(H)** corresponds to confirming a hypothesis with device models. Confirmation can be viewed as the following implications:

$$H \rightarrow C_1$$
$$\ldots$$
$$H \rightarrow C_n$$

If F is true, but **consistent(H)** is false because **not**(C_i) is true, then the diagnosis heuristic can revised to:

F and C_i and consistent(H) → H

In some cases, checking the consistency of a hypothesis with the device models is more properly viewed as the following implication:

B and H → C_i

The situation when **consistent(H)** is false because B is true and C_i is false corresponds to the case that the revised heuristic is used in addition to the old heuristic. The form of the revised heuristic in this case is:

F and C_i and B and consistent(H) → H

The point of failure-driven learning of diagnosis heuristics is that it is simpler to rule out a hypothesis by testing for C_i than proving **consistent(H)**.

An Example

In this section, we describe an example of how the performance of the expert system to diagnose faults in the attitude control system is increased through failure-driven learning. To follow this example, it is necessary to know a little about attitude control.

Attitude Control

The attitude control system consists of a number of sensors which calculate the satellite's orientation on the three axes (called yaw, pitch and roll) by detecting the location of the earth and the sun and a set of reaction wheels which can change the satellite orientation if it deviates from the desired orientation due to torques such as solar pressure. There are four reaction wheels (PY+, PY-, PR+, and PR-), arranged on the four sides of a pyramid (see Figure 1). Pitch momentum is stored as the sum of all four wheel speeds; roll momentum is stored as the difference between the PR+ and PR- speeds; and yaw momentum is stored as the difference between the PY+ and PY- speeds.

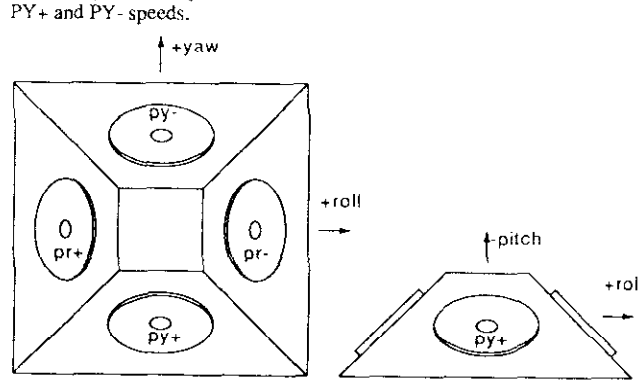

Figure 1: The reaction wheels

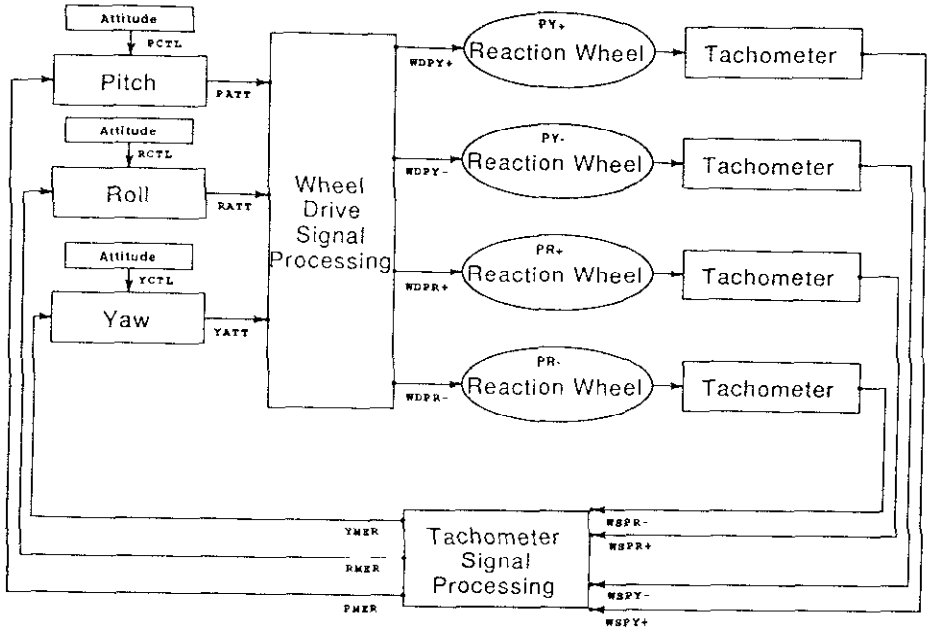

Figure 2: Block diagram of the attitude control system

A diagram of the attitude control system appears in Figure 2. The signals YATT, RATT, and PATT represent the attitude on the yaw, roll and pitch axes respectively. The wheel drive signal processing component issues drive signals to the motor of the reaction wheels to change the wheel speeds to correct for any deviations from the desired attitude. The wheel drive signals are WDPY+, WDPY-, WDPR+ and WDPR- for the PY+, PY-, PR+ and PR- wheels respectively. The wheel speeds are measured by tachometers yielding the signals WSPY+, WSPY-, WSPR+ and WSPR-. The tachometer signal processing module converts the four wheel speeds to the three values called momentum equivalent rates (YMER, RMER, and PMER) representing the equivalent wheel speeds on the three axes. These equivalent wheel speeds are also combined with the attitude information from the sensors to yield the estimated attitudes (YATT, RATT, and PATT).

The attitude control system contains the logic necessary to maintain the desired attitude. For example, to compensate for a disturbance on the roll axis, the difference between the speed of PR+ and PR- wheels must change. Once or twice each day, at a predetermined part of the satellite's orbit if the wheel speeds exceed a certain speed, the momentum stored by the wheels is dumped by firing a thruster.

ACES: The Attitude Control Expert System

One reason that our particular satellite was chosen for this research is that The Aerospace Corporation possesses a simulator for the attitude control system which generates telemetry tapes reflecting faulty behaviors to aid engineers in faults diagnosis. In addition, these tapes serve as input to our expert system. ACES consists of two major modules:

- *Monitor*. This module converts the raw telemetry data to a set of features which describe the atypical aspects of the telemetry. In ACES, the features detected include:
  - *(value-violation signal start-time end-time value)*: Between *start-time* and *end-time* the average *value* of *signal* has taken on an illegal *value*.
  - *(jump signal start-time end-time amount start-value end-value slope)*: The *signal* has changed from *start-value* to *end-value* between *start-time* and *end-time*. *Amount* is the difference between *start-value* and *end-value* and *slope* is *amount* divided by the difference between *start-time* and *end-time*.
- *Diagnostician*. This module finds an explanation for the atypical features.

In this article, we focus on the learning in the diagnostician. The diagnostician is comprised of several cooperating modules:

- *Fault Identification*. The atypical features are used as symptoms of faults by heuristic rules to postulate a hypothesis which could account for the behavior of the satellite.
- *Fault Confirmation*. This step compares the actual device functionality to the functionality as specified by a device model. This process either can confirm or deny that a hypothesized fault is present. If a hypothesis is denied, an attempt is made to identify another fault.
- *Fault Implication Analysis*. After a fault has been confirmed, the effect of the fault on the values of other telemetry signals is assessed. A model of the attitude control system predicts the values of telemetry signals which might be affected by the fault. The predicted telemetry values are analyzed by the monitor to see if they are atypical.

  Descriptions of atypical predicted values are then compared against the set of atypical features to explain any features which are a result of a confirmed fault.

Refining Fault Diagnosis Heuristics

For this example, the initial fault diagnosis heuristics are quite simple. Figure 3 presents the definition of three fault diagnosis rules. These PROLOG rules have a LISP-like syntax since our PROLOG is implemented in LISP. The first element of a list is the predicate name and all variables are preceded by "?". The part of the rule preceded by ":-" is a fault hypothesis, and the part of the rule after ":-" are those conditions which are necessary to be proved to propose the hypothesis. These rules implement three very crude diagnosis heuristics: "if the speed of a reaction wheel is 0, then the tachometer is broken", "if the speed of a reaction wheel is 0, then the wheel drive is broken" and "if there is a change of momentum, then a thruster has fired to unload the momentum".

Telemetry data indicating the values of the momentum equivalent rates, attitude and wheel speeds are illustrated in figure 4. Typical values for these signals are present between 1:07 and 1:08. After 1:08 the monitor notices several atypical features:

1. YMER, RMER, and PMER have changed an unusual amount.
2. WSPR+, WSPR-, WSPY+, and WSPY- have changed an unusual amount.
3. YATT, PATT, and RATT have changed an unusual amount.
4. WSPR+ is 0.

```
1: (problem (problem wheel-tach ?from
    (broken-wheel-tach ?wheel ?from)))
     :-
;there is a tachometer stuck at 0
(feature(value-violation ?sig ?from
                         ?until 0))
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
;if the speed of a wheel is 0

2: (problem (problem wheel-drive ?from
    (broken-wheel-drive ?wheel ?from
                        ?sig)))
     :-
;there is a wheel drive motor not
;responding to the drive signal
(feature(value-violation ?sig ?from
                         ?until 0))
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
;if the speed of a wheel is 0
```

```
3: (problem (normal-operation ?device
              ?from-jump ?end-jump
    (wheel-unload ?axis ?sign ?thruster
              ?ntimes ?jump)))
     :-
;there is a wheel unload on the ?axis
;in the ?sign direction
(isa  ?sig
      momentum-equivalent-rate-signal)
(feature (jump ?sig ?from-jump ?end-jump
               ?jump ?start ?end ?slope))
;if there is a jump of the of
;YMER, RMER, or PMER
(momentum-sig-axis ?sig ?axis)
(is ?s (sign ?jump))
(opposite ?s ?sign)
(thruster-axis ?device ?axis ?sign)
;find the thruster on the same axis
;as the momentum change in the
;opposite direction
```

Figure 3: Initial Fault Diagnosis Heuristics

Since ACES is implemented in PROLOG, it tries the heuristic rules in the order that they are defined. However, for the purposes of learning, the ordering of the rules undefined. (This is implemented by randomly changing the order of the rules before each run.) This prevents one heuristic from relying on the fact that another fault proposed by an earlier heuristic has been ruled out.

In this training example, the third rule in Figure 3 first hypothesizes that the change in PMER is due to a wheel unload. The device model of a thruster reveals that there are two enabling conditions for a wheel unload. First, the satellite must be in part of the orbit called the wheel unload window. Second, the satellite must be in a high momentum state. In this example, the satellite is in a high momentum state, but it is not in the unload window. Therefore, the hypothesis is denied.

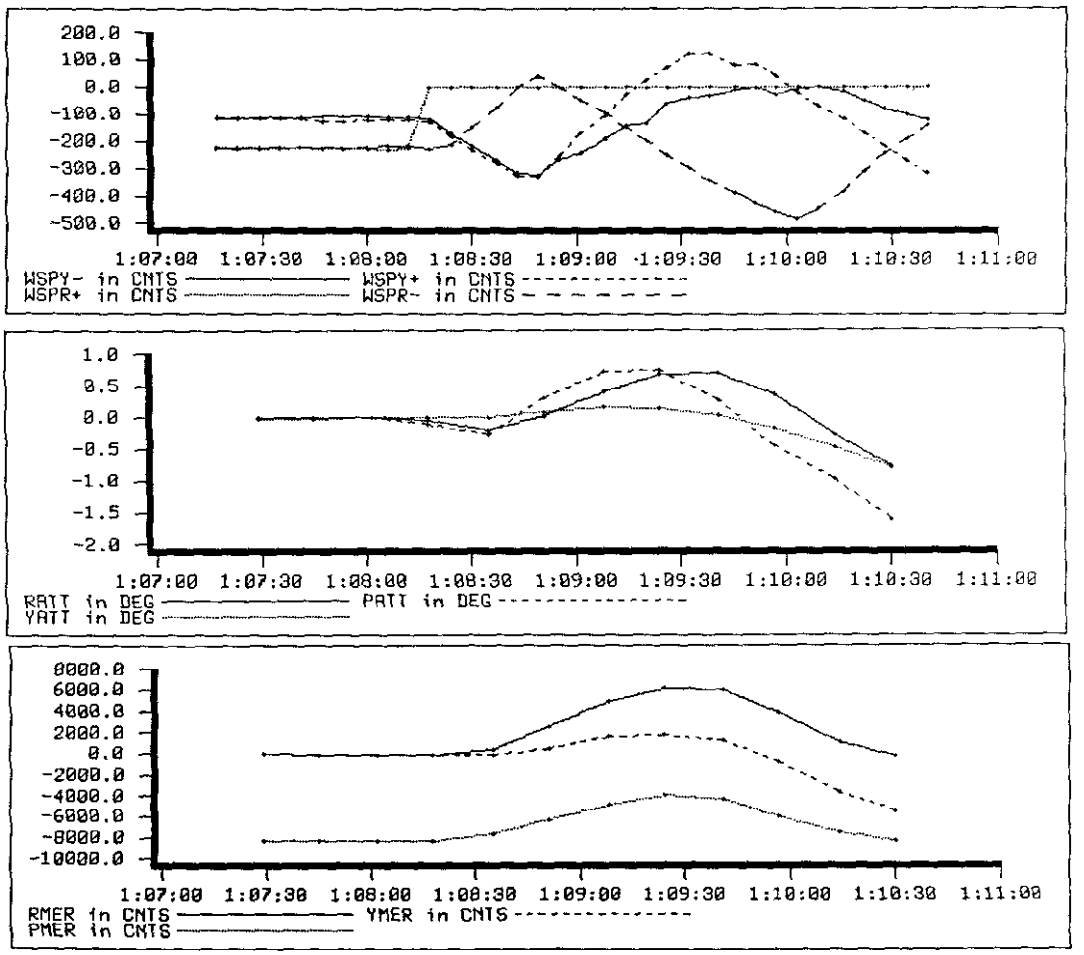

Figure 4: Telemetry data after a broken tachometer

The hypothesis failure is found to be caused by not considering one of the enabling conditions of the wheel unload (Hypothesized Unusual Mode-Enablement Violated). The heuristic is revised to include this enabling condition (see Figure 5). Since the heuristic and the explanation for the failure both applied to thrusters, the revised rule replaces the old version. In addition to checking the enabling condition, the revision includes a call to the predicate "cache-proved" which indicates that if this rule succeeds, there is no need to recheck the enabling condition "unload-window-status" during the confirmation process.

```
(problem (normal-operation ?device
                ?from-jump ?end-jump
     (wheel-unload ?axis ?sign ?thruster
                ?ntimes ?jump)))
     :-
(IN-MODE WHEEL-UNLOAD-WINDOW ?START-97 ?END-98)
;MAKE SURE THE SATELLITE IS IN THE UNLOAD WINDOW.
(isa  ?sig
      momentum-equivalent-rate-signal)
(feature (jump ?sig ?from-jump ?end-jump
               ?jump ?start ?end ?slope))
(AFTER ?FROM-JUMP ?START-97)
;MAKE SURE THE JUMP STARTS AFTER
;ENTERING THE UNLOAD WINDOW.
(BEFORE ?END-JUMP ?END-98)
;MAKE SURE THE JUMP ENDS BEFORE
;EXITING THE UNLOAD WINDOW.
(momentum-sig-axis ?sig ?axis)
(is ?s (sign ?jump))
(opposite ?s ?sign)
(thruster-axis ?device ?axis ?sign)
(CACHE-PROVED UNLOAD-WINDOW-STATUS)
```

Figure 5: Revised Wheel Unload Heuristic– changes in SMALL CAPITALS

This example also illustrates another point. We are not drawing an arbitrary line between what we call generating hypotheses and confirming hypotheses. Not all information is moved from confirmation to generation. Rather, we move only those tests from confirmation which prevent the generation of an erroneous hypothesis. In this example, the high momentum state information is not included in the heuristic because it does not differentiate a wheel unload from the true fault.

After the heuristic has been revised, diagnosis continues. The next hypothesis is a failure of a tachometer of the PR+ reaction wheel. This hypothesis is proposed by the second rule in Figure 3. The device model confirms this hypothesis.

One further example will help to illustrate the other strategies for revising fault diagnosis heuristics. Figure 6 contains the relevant telemetry data. For this telemetry tape, the monitor notices several atypical features:

1. WSPR-, WSPR+, WSPY+ and WSPY- have changed an unusual amount.
2. WSPR+ and WSPR- are 0.

The first hypothesis proposed by the first rule in Figure 3 is that the tachometer of the PR- wheel is stuck at 0. The confirmation module denies this hypothesis for the following reason: if the tachometer were stuck at 0, the attitude of the satellite would change drastically. (The attitude control system would believe that the wheel was not storing any momentum when in fact it is. To compensate for the erroneous report of loss of momentum, the attitude control system would adjust the momentum of the other wheels, changing the attitude of the satellite.) Since the attitude did not change, the heuristic must be revised to avoid the generation of this hypothesis in future similar cases. The hypothesis failure is caused by not checking the implications of a faulty tachometer (Hypothesized Fault- Inconsistent Prediction). Checking any of the attitude signals would suffice to distinguish a faulty tachometer from the actual fault. In Figure 7, the revision tests YATT.

After the heuristic has been revised, diagnosis continues. The next hypothesis proposed by the second rule in Figure 3 is that the wheel drive of the PR- wheel is broken. The device model of a wheel drive includes the following information: the wheel speed is proportional to the integral of the wheel drive signal. If the wheel drive signal is positive, the wheel speed should increase.

During the time that WSPR- increased from -100 to 0, WDPR- was positive (see figure 6). Therefore, the PR- wheel was not ignoring its drive signal and the hypothesis is denied. The hypothesis failure is caused by the fact that WSPR- wheel is indeed doing something very unusual by changing so rapidly and stopping. However, it is doing this because it is responding to WDPR-. The heuristic which proposed this fault is revised to consider the functionality of the device (Hypothesized Fault– Unusual Input).

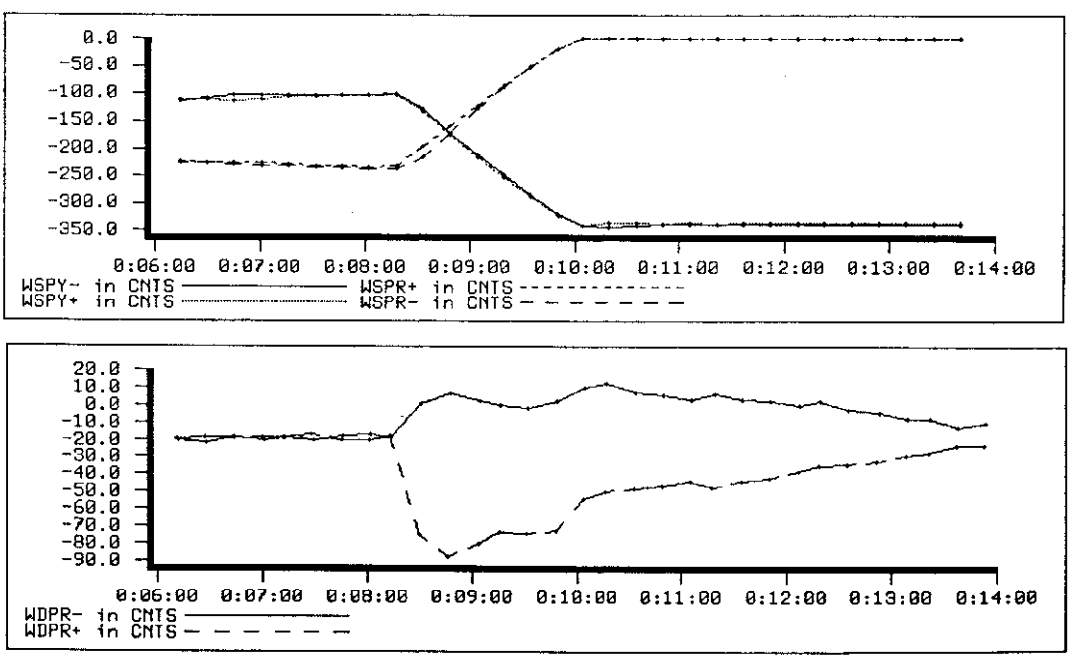

Figure 6: Telemetry data after a broken wheel drive

```
(problem (problem wheel-tach ?from
   (broken-wheel-tach ?wheel ?from)))
   :-
(FEATURE (VALUE-VIOLATION YATT ?FROM-32
                           ?END-33 ?VALUE-34))
;MAKE SURE THE YAW ATTITUDE HAS BEEN DISTURBED
(feature(value-violation ?sig ?from
                         ?until 0))
(AFTER ?FROM-32 ?FROM)
;MAKE SURE THE ATTITUDE DISTURBANCE IS
;AFTER THE VALUE VIOLATION
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
(CACHE-PROVED ATTITUDE-DISTURBANCE)
```

Figure 7: Revised Faulty Tachometer Heuristic– changes in SMALL CAPITALS

In Figure 8, the revised heuristic checks that the change of the wheel speed as it approaches 0 is not due to the drive signal. Since our heuristic rules and our device models are implemented in the same language, it is possible to move code from the device model to a heuristic rule by renaming variables. In other systems, this may not be possible. However, this strategy would still apply if the rule could be revised to indicate what part of the device model to check for (e.g., test that the observed wheel speed could not be produced given the wheel drive between $time_1$ and $time_2$). In ACES, it is possible to revise the rule to specify how the test should be performed instead of what test should be performed.

```
(problem (problem wheel-drive ?from
 (broken-wheel-drive ?wheel ?from ?sig)))
   :-
(FEATURE(JUMP ?SIG ?FROM-37 ?UNTIL-38 ?JUMP-39
          ?START-40 ?END-41 ?SLOPE-42))
;THERE IS A CHANGE IN THE WHEEL SPEED
(feature(value-violation ?sig ?from
                         ?until 0))
(AFTER ?FROM ?FROM-37)
;THE WHEEL SPEED REACHES 0 AFTER IT CHANGES
(measurement  ?sig ?wheel speed ?tach)
(isa ?wheel reaction-wheel)
(DRIVES ?DRIVE-43 ?WHEEL)
(MEASUREMENT ?DRIVE-SIGNAL-44 ?DRIVE-43
          AMPLITUDE DIRECT)
;FIND THE WHEEL DRIVE SIGNAL OF THE ?WHEEL
(IS ?DRIVE-SIGNAL-SIGN-45
    (TELEMETRY-SIGNAL-SIGN ?DRIVE-SIGNAL-44
                           ?FROM-37 ?UNTIL-38))
;FIND THE SIGN OF THE THE DRIVE SIGNAL
;DURING THE JUMP
(IS ?SLOPE-SIGN-46 (REPORT-SIGN ?SLOPE-42))
;FIND THE SIGN OF JUMP
(NOT (AGREE ?SLOPE-SIGN-46 ?DRIVE-SIGNAL-SIGN-45))
;MAKE SURE THE DIRECTION OF THE JUMP
;DISAGREES WITH THE DRIVE-SIGNAL.
(CACHE-DISPROVED WHEEL-DRIVE-STATUS)
```

Figure 8: Revised Wheel Drive Heuristic– changes in SMALL CAPITALS

After the heuristic has been revised, another hypothesis is found to account for the atypical features: the faulty wheel drive heuristic proposes that the PR+ drive is ignoring its input since WSPR+ is 0, and when it increased to 0, WDPR+ was negative indicating that the speed should decrease (see Figure 6). The confirmation of this hypothesis is trivial since the heuristic already proved that the drive was not functioning according to its device description. After the fault is confirmed, the effects on the rest of the attitude control system are assessed. Since roll momentum is stored as the difference between the speed of the PR+ and PR- reaction wheels, when WSPR+ goes to 0, WSPR- should change by the same amount. The satellite was in a very unusual state prior to the failure: WSPR+ and WSPR- were equal. When the PR+ drive broke, WSPR- went to 0 to compensate for the change in WSPR+. In addition, since the pitch momentum is stored as the sum of all four wheels, to maintain pitch momentum WSPY+ and WSPY- decreased by the amount that WSPR+ and WSPR- increased. While WSPY+ and WSPY- decreased, the difference between them remained constant to maintain the yaw momentum. The broken PR+ wheel drive accounts for the atypical features and the diagnosis process terminates.

Results

There are two standards for evaluating the effects of learning in ACES. First, there is the performance of ACES using the rules in Figure 3. We call this version naive-ACES. Additionally, there is the performance of ACES using rules hand-coded from information provided by an expert. We call this version of the system expert-ACES. The performance of the naive-ACES after learning is compared to naive-ACES and expert-ACES in Figure 9 and Figure 10. There are four test cases which are used for comparison:

1. A tachometer stuck at 0 (see Figure 4).
2. A wheel drive ignoring its input when the opposite wheel is at the same speed (see Figure 6).
3. A wheel unload (i.e., the speed of the reaction wheels is changed by the firing of a thruster).
4. A wheel drive ignoring its input in the usual case where the opposite wheel is at a different speed.

The data in Figure 9 demonstrate that the failure driven learning technique presented in this paper improves the simple fault diagnosis heuristics to the extent that the performance of ACES using the learned heuristics is comparable to the system using the rules provided by an expert. In one case, the performance of the learned rules is even better than the expert provided rules. This particular case is the previous example in which a wheel drive broke when the satellite was in an unusual state. The heuristic provided by the expert did not anticipate the rare condition that two opposing wheel speeds were equal.

CASE	fault	naive ACES	naive+ learning	expert ACES
1	tachometer	21	1	1
2	wheel drive	4	1	2
3	wheel unload	1	1	1
4	wheel drive	2	1	1

Figure 9: Number of Fault Hypotheses

The data in Figure 10 reveal that the number of logical inferences required by the expert system decreases after learning. This demonstrates that after learning, the expert system is doing less work to identify a failure rather than moving the same amount of work from hypothesis confirmation to hypothesis generation. Comparing the number of inferences required by naive-ACES after learning to those of expert-ACES is not actually fair since it appears that the expert's rules at times test some information retested by the confirmation process. Recall that retesting is avoided by a revised rule since the revision contains information to cache the results of consulting a device model. It has been our experience that this cache reduces the number of inferences by approximately ten percent. An additional ten percent of the inferences are saved through intelligent ordering of clauses of revised rules compared to our initial simple approach of appending the revision to the end of a rule.

CASE	fault	naive ACES	naive+ learning	expert ACES
1	tachometer	2268	211	584
2	wheel drive	1238	616	910
3	wheel unload	870	861	947
4	wheel drive	745	409	643

Figure 10: Number of Inferences to Generate and Confirm Fault

Conclusion

We have presented an approach to learning fault diagnosis heuristics by determining what aspect of a device model must be consulted to distinguish one fault from another fault with similar features. This approach relies on explaining why a heuristic does not apply in a certain case and correcting the heuristic to avoid proposing an erroneous fault hypothesis. Applying this technique to a simple version of the ACES expert system for the diagnosis of faults in the attitude control system yields performance comparable to and in some cases better than the performance of ACES with expert fault diagnosis heuristics.

Acknowledgements

Comments by Anne Brindle, Jack Hodges, Steve Margolis, Rod McGuire and Hilarie Orman helped clarify this article. This research was supported by the U.S. Air Force Space Division under contract F04701-85-C-0086 and by the Aerospace Sponsored Research Program.

References

[1] Cantone, R., Pipitone, F., Lander, W., & Marrone, M. Model-based Probabilistic Reasoning for Electronics Troubleshooting. In *Proceedings of the Eigth International Joint Conference on Artificial Intellegence*, pages 207-211. IJCAI, Vancouver, August, 1983.

[2] Charniak, E., Riesbeck, C. and McDermott, D. *Artificial Intelligence Programming*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

[3] Davis, R., Shrobe, H., *et al.* Diagnosis Based on Description of Structure and Function. In *Proceedings of the National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, Pittsburgh, PA, 1982.

[4] de Kleer, J. & Brown, J. A Qualitative Physics Based on Confluences. *Artificial Intelligence* 24(1), 1984.

[5] DeJong, G. Acquiring Schemata Through Understanding and Generalizing Plans. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. Karlsruhe, West Germany, 1983.

[6] Genesereth, M., Bennett, J.S., Hollander, C.R. DART: Expert Systems for Automated Computer Fault Diagnosis. In *Proceedings of the Annual Conference*. Association for Computing Machinery, Baltimore, MD., 1981.

[7] Michie, D. Inductive Rule Generation in the Context of the Fifth Generation. In *Proceedings of the International Machine Learning Workshop*. Monticello, Illinois, 1983.

[8] Mitchell, T. Generalization as Search. *Artificial Intelligence* 18(2), 1982.

[9] Mitchell, T., Kedar-Cabelli, S. & Keller, R. *A Unifying Framework for Explanation-based Learning*. Technical Report, Rutgers University, 1985.

[10] Naish, Lee. Prolog Control Rules. In *Proceedings of the Ninth International Joint Conference on Artificial Intellegence*, pages 720-722. IJCAI, Los Angeles, CA, August, 1985.

[11] Nelson, W.R. REACTOR: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents. In *Proceedings of the National Conference on Artificial Intelligence*. AAAI, Pittsburgh, PA, 1982.

[12] Schank, R. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.

[13] Sembugamoorthy, V. & Chandraskaran, B. *Functional Representation of Devices and Compilation of Diagnostic Problem Solving Systems*. Technical Report, Ohio State University, March, 1985.

[14] Shortliffe, E.H. *Computer-based Medical Consultation: MYCIN*. American Elsevier, New York, NY, 1976.

[15] Stallman, R. M. & Sussman, G. J. Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. *Artificial Intelligence* 9(2):135-196, 1977.

[16] Vere, S. Induction of Concepts in the Predicate Calculus. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*. Tbilisi, USSR, 1975.