N93-11925

A STATE-BASED APPROACH TO TREND RECOGNITION AND FAILURE PREDICTION FOR THE SPACE STATION FREEDOM

Kyle S. Nelson
Research Scientist
Honeywell Systems and Research Center
3660 Technology Dr.
Minneapolis, MN 55418

George D. Hadden, Ph.D.
Research Fellow
Honeywell Systems and Research Center
3660 Technology Dr.
Minneapolis, MN 55418

1.0 ABSTRACT

A state-based reasoning approach to trend recognition and failure prediction for the Attitude Determination, and Control System (ADCS) of the Space Station Freedom (SSF) is described. The problem domain is characterized by features (e.g. trends and impending failures) that develop over a variety of time spans, anywhere from several minutes to several years. Our state-based reasoning approach, coupled with intelligent data screening, allows features to be tracked as they develop in a time-independent manner. That is, each state machine has the ability to encode a time frame for the feature it detects. As features are detected, they are recorded and can be used as input to other state machines, creating a hierarchical feature recognition scheme. Furthermore, each machine can operate independently of the others, allowing simultaneous tracking of features. State-based reasoning was implemented in the trend recognition and the prognostic modules of a prototype Space Station Freedom Maintenance and Diagnostic System (SSFMDS) developed at Honeywell's Systems and Research Center.

2.0 SPACE STATION APPLICATION

The Space Station Freedom Maintenance and Diagnosis System (SSFMDS) project was established as a feasibility study whose purpose was to illustrate how Expert Systems could augment the Fault Detection, Isolation, and Recovery (FDIR) of Freedom's Attitude Determination and Control System (ADCS). The ADCS comprises four subsystems, called Orbital Replaceable Units (ORU), three of which are Honeywell's responsibility: the Star Trackers (ST) and Inertial Sensor Assemblies (ISA) for attitude determination and the Control Moment Gyroscopes (CMG) for attitude control. (The other subsystem is called the Reaction Control System and is a set of hydrazine jets also used for attitude control.)

SSFMDS comprises a set of two cooperating expert systems, one running on Freedom and the other running either on Freedom or on the ground. Since the onboard expert system is in closer contact with the environmental sensors, we call it the "On-line" system. The other, of course, is called the "Off-line" system and receives its information through the On-line system. When running in the actual Freedom environment, the two expert systems will communicate over the main telemetry link.

In our prototype, we demonstrate the operation of the On-line and Off-line systems by running each on a separate machine. We have the ability to simulate the telemetry link using either a serial RS232 or an ethernet connection. The prototype currently runs in a Unix/ XWindows/Common Lisp environment and supports both IBM PS/2 Model 70 portables and Sun workstations in any combination. One of the reasons that we chose the IBM as a prototype host is that it contains the processor (the Intel 80386) that has been chosen to be the heart of Freedom's Standard Data Processor (SDP).

An area that we do not cover in detail in this paper but that nevertheless deserves some mention is that of data filtering techniques. We have extended methods described in [Washington and Hayes-Roth, 1989] to significantly reduce the bandwidth required to communicate health monitoring and other ADCS data between the On-line and Off-line systems. Traditionally, bandwidth on spaceport telemetry links has been very tight -- there is never enough to go around and Space Station Freedom is no exception. It uses these techniques (including, for instance, dynamic thresholding) to send only the necessary data between the On-line and Off-line systems.

SSFMDS has three areas of expertise: predictive maintenance, diagnosis, and maintenance aiding. Of the three, the first has received the least attention in the more traditional Space Station Freedom software and thus has been the area where we have concentrated most of our efforts. Techniques we have developed for predictive maintenance, including trend recognition and failure prediction, form the main topic of this paper.

2.1 Why not use model-based reasoning?

A word about model-based reasoning: although we have built a number of expert systems, many of which have been model-based, we found difficulty in applying model-based techniques to the domain of predictive maintenance on Freedom. One problem is that there are conditions we predict which have no known physical model. One example is the degradation of the mirror in the ISA lasers. There is a relationship between the current/output power curves such that prediction of a laser failure is possible several months away. No one knows why this relationship exists, which to say the least, makes it difficult to model. Cases like this have caused us to look for other methods.

3.0 STATE-BASED FEATURE RECOGNITON

One method we found, and the subject of this paper, is to represent trends and predictive scenarios in the ADCS as state machines. This technique, called State-Based Feature Recognition (SBFR), will be discussed in the following paragraphs.

3.1 Feature Representation

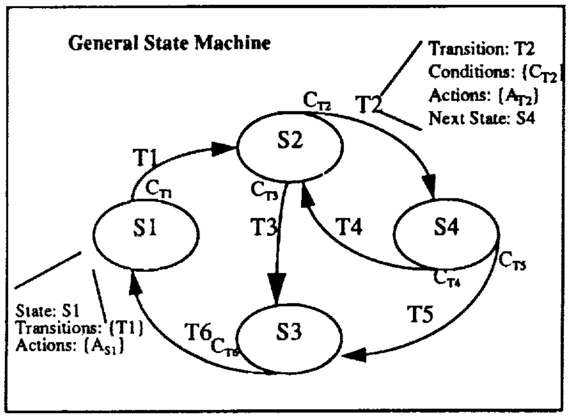


Figure 1 Sample State Machine.

Features to be recognized using SBFR are represented as finite state machines, with each feature having its own state machine. The following refers to Figure 1 which illustrates an example of a generic state machine. The state machine is made up of a set of states, {S1,...,S6} and transitions between those states, {T1...T6}. Each state in the machine represents a stage in the identification of the feature. Each state has associated with it a set of transitions and, optionally, an action to be executed when the state is entered, called a state action. Similarly, each transition may, optionally, have an action associated with it, called a transition action. State actions are useful if entering a state means something no matter how it was entered. For example, suppose that entering state S3 implied a dangerous engine condition, a state action would handle this no matter how the state was entered. Transition actions, on the other hand, are useful in situations where entering a state means something different depending on the transition traversed. For example, in Figure 1, a notification may necessary when entering state S3 from state S2, while no action is needed when entering state S3 from state S4.

Each transition has a set of conditions associated with it. The conditions define when the transition may be traversed. In other words, when the condition for a particular transition evaluates to true, the machine can traverse the transition. For example, in Figure 1, C_{T2} is the condition for T2, if the current state is S2 and C_{T2} becomes true, S4 will become the next state. When that occurs, the actions defined for T2, {A_{T2}} will be executed. If S4 as any state actions associated with it they will also be executed.

The machines described above have their roots in automata theory. State machines that specify actions on the transitions are called Mealy machines and those whose actions are specified on the states are called Moore machines. From automata theory, any Moore machine has an equivalent Mealy machine and, conversely, any Mealy machine has an equivalent Moore machine [Hopcroft and Ullman, 1979]. Thus, only one type of action is required. Transition actions can be replaced by adding more states and associating actions with those, while state actions can be replaced by defining the action on all of the transitions entering the state. Eliminating either type of action, however, has certain drawbacks when used in SBFR. A transition action, replaced by a state action, increases the complexity of the machine by adding more states to the machine, while a state action replaced by transition actions will make the machine harder to maintain. That is, instead of maintaining one action for the state, actions must be maintained for all of the transitions. Consequently, supporting both types of actions is recommended to preserve the intuitive nature of the machines.

During operation, only one state of the machine is active, called the current state. The transitional conditions are, therefore, typically mutually exclusive to ensure that two state transitions cannot simultaneously be true. Of course, there may be circumstances when it is reasonable to have a machine be in two states at once, and state machines can be implemented to support that, but for the applications discussed below, it is more intuitive to limit the machine to one currently active state. This paper assumes that the transitions for given state are mutually exclusive and that there is only one currently active state per machine.

The semantic meaning of a feature is captured by the states and transitions, while the specifics of the feature (i.e. the exact data which causes the state machine to move from one state to another) are captured by the transitional conditions. This results in a separation between the general definition of a feature and its real world implementation, allowing a general machine to be instantiated in many different contexts. This, in effect, permits the construction of a library of feature descriptions that can be instantiated for many different applications by only changing the specifics of the transitional conditions. For instance, an increasing trend in computer CPU utilization and daily air temperature could have the same general definition (i.e. the same states and transitions), but different transitional conditions. The CPU trend will have a much smaller time span and magnitude than the temperature trend. Since these trends are so similar, the state machines will be identical except of the magnitude and duration of the feature.

3.2 SBFR Application Characteristics

Each state machine moves from one state to another in a well-defined order that depends on which transitional conditions evaluate to true. Thus, features recognized by a well-defined order of stages are those best represented and recognized using SBFR. The stages are usually ordered temporally, although other orderings may also be possible. As will be discussed later, in section 4.0 and section 5.0, failure prediction and trend identification are features whose stages are temporally ordered.

One case where the features are typically not well-ordered is fault diagnosis. Faults are typically recognized by the presence of a set of symptoms that appear at the time of the failure. These symptoms usually do not appear in any well-defined order (if they did then a state machine could be used as a means to predict the failure), resulting in a state machine of only two states, essentially an if-then statement. So, while fault diagnosis could certainly be implemented using SBFR, another approach would probably be better.

3.3 SBFR Is Data-Driven

The fact that SBFR has a natural applicaticn to features recognized by a well-defined sequence of events implies that SBFR should execute in a data-driven manner. That is, the SBFR module is invoked when new data enters the system. The transitional conditions associated with the current state are evaluated using the new data and any required historical data. If a transition's condition is true, the machine will move to the next state, executing any actions defined on the transition and the new state entered.

The data-driven nature of state machines allows features to be detected concurrently. The same data can be used for any number of state machines, allowing a system implementing SBFR to simultaneously track several features in the incoming data. In many situations, particularly when real-time data analysis is required, the ability to track many features in parallel is not only useful but necessary.

The following sections will illustrate in greater detail how SBFR is implemented to recognize features in parameter data. Two implementations of SBFR are present in the SSFMDS, trend recognition and failure prediction. Both implementations are based on the general machines discussed above.

4.0 TREND RECOGNITION

One important implementation of SBFR is the difficult, but important, task of trend recognition. Trends can indicate many things about a particular device including impending failures, environmental changes, and other anomalies. It often takes an expert to decipher trends and speculate as to their meaning, especially since the significance of trends is dependent on many factors (e.g. the feature's time span, the parameter's expected behavior, etc.). For instance, fluctuating data for a generally stable parameter would indicate a problem, while the same data for an erratic parameter would be considered normal behavior. Humans are very good at analysis involving pattern detection in noisy, convoluted data. Domain experts are even better at this sort of analysis, since they know what parameter trends are significant to the monitored device. Experts are, however, a scarce commodity and their time is too valuable to watch data scroll by on a screen. Furthermore, some trends may occur too rapidly for a human to detect it, some significant trends may happen in milliseconds. These two problems have lead us to search for ways to automate the process using computers. The problem, however, is programming the computer to both recognize significant parameter trends and ignore the insignificant ones. SBFR provides a representation scheme encompassing an intuitive method for describing the trend in human terms and an easy way to translate it into computer terms.

4.1 SSFMDS Implementation

The state machines implemented to recognize trends are cleverly called trend machines. Each trend machine is built on the principles discussed in section 3.0, with the addition of an initial state and a final state. The initial state is the default starting state of the machine and represents no trend being detected, i.e. no evidence of the trend has been seen. The final state is just the opposite, entering this state indicates that a trend has been identified. Like the general state machines, a trend machine will remain in a state until incoming data causes it to transition to the next state.

The final state, denoted by a double circle in Figure 2 and Figure 3, is an important state of the trend machine. When the machine enters this state, a trend has been detected and a notification is sent to the knowledge base, along with the information collected as the trend was being detected. As long as the machine remains in the final state, the trend notification is updated with the current information, allowing the SSFMDS to monitor long-term trends with a minimal amount of effort.

A set of trend machines is associated with each parameter and uses data local to that parameter as input. A global view encompassing the entire ADCS could be provided, but would dilute the intended function, which is to provide information about ORU parameters to the SSFMDS reasoning mechanisms at a higher level than raw data. Features dependent on several ORU parameters do exist and are considered in the discussion of failure prediction, see section 5.0.

Since trend machines are associated with a single parameter, they can be executed efficiently. When new data enters the SSFMDS, the only transition condition functions evaluated are those associated with the current states of the parameter's machines. A global view would require evaluating the transition functions of all trend machines. This can be significant when the system being monitored has several hundred parameters, each of which has several trends identified for it.

A parameter's trend machines are implemented hierarchically. Filtered data from the on-line system is input into the first trend machine layer. The machines at this level recognize simple trends like increases, decreases, spikes, etc. Trends of this nature only require raw data as input. These trends are used as input into the next layer of trend machines which can recognize more complex trends. Currently, the SSFMDS trend recognition module implements the first layer of the hierarchy and has the machinery to implement higher layers. This hierarchy enables complex trends to be broken down into components consisting of simpler trends. The following two examples will illustrate trend machines at the first and second level of the hierarchy.

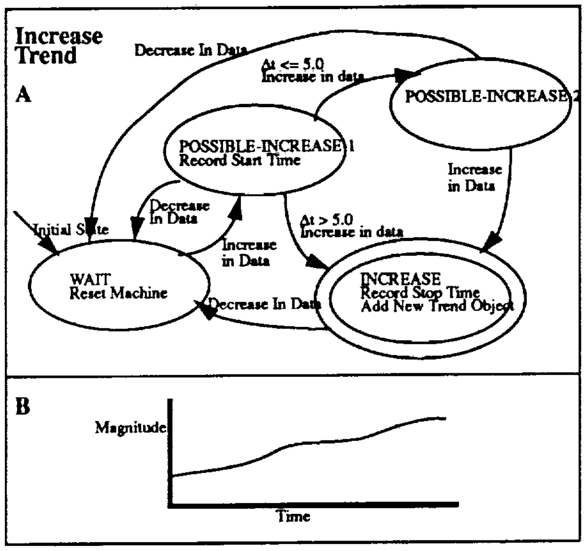


Figure 2 Simple Increase Trend and Sample Data

Figure 2 A shows a trend machine used to recognize a simple increase trend, like the one illustrated in Figure 2 B. Note that this is just one example, the transitions and states defined for this machine could be changed to suit any situation. In this case, an increase is defined as being two jumps in the data occurring greater than 5 time units apart, or three jumps in the data if the first two jumps happen in less than or equal to 5 time units. The reason for the time limit is to differentiate between an increase and a spike. Anytime a decrease in the data is seen, the machine will transition back to the initial state. When the machine enters the final state (denoted by a double circle in the diagram) the knowledge base is informed that an increasing trend has been identified and is sent the information collected during the trend identification (the start and stop times in this case).

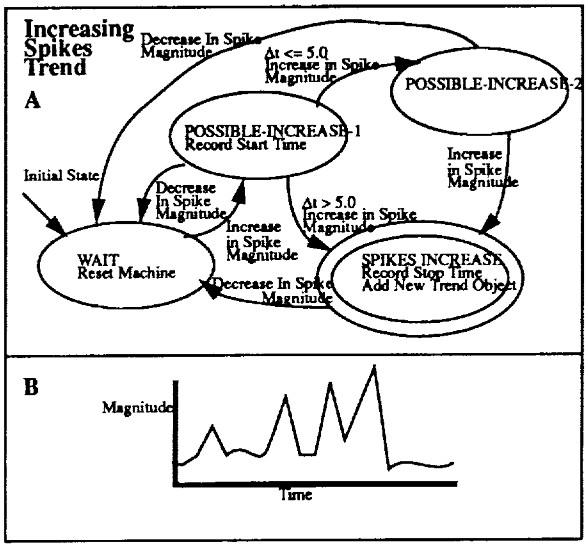


Figure 3 Increasing Spikes Trend and Sample Data.

Figure 3 shows a trend machine defined at the second level of the hierarchy. In this case, the machine is used to recognize an increasing-spikes trend, i.e. a trend of spikes with increasing magnitude, like the one in Figure 3 B. This trend is at the second level of the hierarchy because it takes simple trends (spikes in this case) as input and outputs trends made up of those simple trends. When a spike is noticed by the trend recognition module, it is used as input into the machines at the next level where the spike magnitude is used in the transitional conditions. Notice that the only difference between the machines of Figure 2 and Figure 3 is that the transitional conditions are different, the states are otherwise identical. In Figure 2 the conditions use the value of the incoming filtered data, but in Figure 3 the magnitude of detected spike trends is used. This illustrates how one basic trend machine can be applied in different circumstances. The trends detected provide information to personnel monitoring the ADCS as well as information to other modules of the SSFMDS, specifically the failure prediction module.

5.0 FAILURE PREDICTION

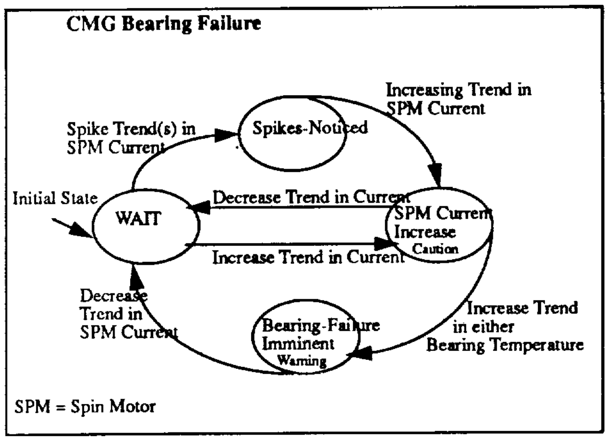
Predicting when a failure will occur is a critical factor in the maintenance of the ADCS. The ADCS is characterized by functionally independent ORUs, low failure rates, and extremely high replacement and repair costs. Spare parts not stored onboard the station must be flown up by space shuttle or rocket, and, regardless of how the replacement arrived, EVA activity is currently required to replace a failed ADCS ORU. Either of these activities can cost millions of dollars and puts equipment and, more importantly, personnel in danger. Forecasting ORU failures can yield tremendous savings by allowing EVA times to be scheduled to accomplish several tasks and allowing spare parts to be flown up on scheduled flights.

Predicting failures, however, is not a simple task and a general approach is not yet available. Failure prediction relys heavily on the experience of experts and their ability to estimate the condition of the equipment. During our research, we found that experts predict failures by noticing certain features in the data over a period of time. For example, features in the ISA laser output current/power can indicate a failure several months before it occurs, allowing plenty of time for repairs to be made. A state machine can be used to capture this information.

5.1 SSFMDS Implementation

The technique is very similar to that used for identifying data trends, discussed in section 4.0. A state machine is defined for each predictive scenario. As data is made available, it is fed into the state machine. The actions associated with the states and transitions are cautions and warnings indicating the estimated time to failure and recommended actions. The data used by the predictive machines includes raw data and trend information from all parts of the ADCS. That is, a predictive machine may monitor trends across different parameters on one or more ORUs. This differs from the approach taken for trend analysis that only considers data for one ORU. For instance, one predictive machine can monitor trends in the wheel unbalance of all ADCS CMGs. These trends should be roughly equal, if one CMG exhibits a different trend (i.e. it is increasing while the others remain steady) a problem may be indicated.

To make this more concrete, consider an example of a CMG predictive scenario that illustrates how a predictive state machine can combine trends from multiple parameters to predict a failure. This example, a CMG spin bearing failure, is based on a failure that occurred on Skylab, rendering one of its CMGs inoperable. This type of spin bearing failure is characterized by a series of zero or more spin motor current spikes followed by a rise in the spin motor current, shortly after this rise, the spin bearing temperature also increases. If this continues, the CMG wheel will seize and stop. The sequence of symptoms is important, if the same sequence were seen in a different order it may indicate a different problem, or no problem at all. If a different problem were indicated, a separate state machine would have to be created to monitor that problem in parallel with the bearing failure machine.



CMG Bearing Failure
Initial State
WAIT
Spike Trend(s) in SPM Current
Spikes-Noticed
Increasing Trend in SPM Current
Decrease Trend in Current
Increase Trend in Current
SPM Current Increase Caution
Increase Trend in either Bearing Temperature
Bearing-Failure Imminent Warning
Decrease Trend in SPM Current
SPM = Spin Motor

Figure 4 Example Predictive State Machine

The state machine reflecting this behavior is shown in Figure 4. After reading the section on trend analysis, the machine should be fairly self-explanatory. As data enters the system and CMG trends are detected, they are fed into the state machine. For instance, if the machine is currently in the WAIT state, the machine is monitoring the data for either spin motor current spikes, or a spin motor current increase trend. When this happens, it will transition to the next state, executing whatever actions are defined on the transitions.

In this case, if an increase was seen in the spin motor current, a caution would be issued by the system indicating that a bearing failure may occur. Notice that the detection of current spikes does not trigger an action. Current spikes were detected prior to the Skylab failure, but were also detected at different times on other Skylab CMGs that did not fail. The presence of spikes, therefore, would reinforce any subsequent conclusions but are not, themselves, indicative of a bearing failure. In other words, current spikes are used to increase confidence in the prediction, but a prediction is not based solely upon their presence.

6.0 FUTURE DIRECTIONS

This research could be extended in a number of directions.

- Embed the state machines in the ORU's and the sensors. Putting the intelligence at the sensor or the ORU would reduce the bandwidth required to report on problems and potential problems to a trickle, freeing it up for other data.One particular avenue we are exploring in this area is making sensors intelligent by providing them with data filters and trend recognition [Wald, Schoess, and Hadden, 1991].

- Explore the relationship between the data filters and the trend machines. It may be the case that ways can be found in which the two subsystems can cooperate to provide an even higher data rate and/or lower bandwidth than they do now.
- Add higher level trend machines. Making the output of one trend machine available as the input to another one turned out to be a good idea, yet no more than two layers have been tested. Perhaps the logical extension of this idea into more layers of trend machines would have application.
- Implement the machines in a forward chaining rule-based language like CLIPS. CLIPS (both the original and CLIPS-Ada) is emerging as a NASA standard. This, coupled with the facts that a forward chaining language is well-suited to a state-based system and the ease of extensibility such a language affords, make it an attractive choice for an implementation vehicle.
- Learning. A number of trends are currently recognized, yet specification of a complete set of "interesting" trends in any one domain is a difficult task. Machine learning techniques may allow us to generate the ability to recognize these trends with a minimum of human input. A more speculative idea is to use neural networks to perform this learning.

Finally, perhaps the most important future direction is the application of the Maintenance and Diagnosis System to other space related projects. It is clear that these systems must be many times more intelligent than their earth-bound counterparts -- a vessel halfway between the Moon and Mars cannot call a tow truck. At the same time, power and space requirements dictate their own constraints. We must use existing technology and develop new technology to assure that our astronauts' journeys are safe.

7.0 Bibliography

Hopcroft, J. and Ullman, J.
Introduction to Automata Theory, Languages, and Computation
Addison-Wesley, 1979.

Wald, J., Schoess, J. and Hadden, G.
Distributed Health Management Systems for GN&C Applications
1st International Conference for Guidance, Navigation, and Control, 1991
Noordwijk, The Netherlands.

Washington, R. and Hayes-Roth, B.
Input Data Management in Real-Time AI Systems
International Joint Conference on Artificial Intelligence, IJCAI-11
1989 pp. 250-255.