To the Graduate Council:

I am submitting herewith a thesis written by Arthur Louis Qualls entitled "Signal Validation Using Expert System Technology." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Robert E Uhrig

Robert E. Uhrig, Major Professor

We have read this thesis
and recommend its acceptance:

BR Upadhyaya

J. W. Kerlin

B. R. Bergen

Accepted for the Council:

Lew Minkel

Vice Provost
and Dean of the Graduate School

SIGNAL VALIDATION USING EXPERT SYSTEM TECHNOLOGY

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Arthur Louis Qualls

June 1988

ACKNOWLEDGMENTS

The author wishes to express his gratitude to the technicians, operators and engineers at Northeast Utilities, Tennessee Valley Authority and Combustion Engineering who provided the information vital to the completion of this thesis, and in particular to:

R. E. Uhrig, for his advice, contribution,and recommendations which made this thesis possible;

B. R. Upadhyaya, for his encouragement, support and hope;

M. W. Roberts, for his devoted assistance in creating the graphics presented in this thesis;

T.W. Kerlin, for his motivation to always do better; and,

P. F. Pasqua, for trying to teach me to think. I hope someday to be able to as he does.

ABSTRACT

A signal validation and sensor information expert system has been implemented and tested. The expert system interactively provides the user with a list of sensors used to monitor a subsystem of a power plant and answers the user's questions about the characteristics of those sensors.

The expert system also performs an evaluation of a sensor's current output characteristics. The evaluation is fully automated and is designed to mimic an operator's method of signal validation. The evaluation is designed to detect three types of possible sensor anomalies. 1. Incorrect average output. 2. Improper noise level, and 3. incorrect response to a system perturbation. There are two tests used to detect each possibly anomalous condition. One test is a comparison of the specified sensor characteristic against the same characteristic of its redundant sensors. The second test is a prediction of the sensor's expected behavior and comparison with its observed behavior.

The evaluation was tested using operational data from twelve sensors used to monitor level, pressure, primary and secondary coolant flow rates, and hot and cold leg temperatures of a steam generator in a commercially operating nuclear power plant. The results of the testing

successfully demonstrated that the evaluation detected the three possible anomalies that it was designed to, and was able to handle conflicting information from the comparison among redundant sensors and the system specific test used to detect the anomalies.

The software package was written in the computer languages LISP and C, and is implemented on a VAX mainframe computer and a MicroVax workstation.

CONTENTS

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

1.1 Review of Previous Work

Diagnosis of malfunctions in process plants has traditionally been in the domain of the process operator, who relies on training, experience, and reasoning ability to diagnose faults[1]. Much work in the development of operator aids for fault diagnosis has focused on detecting and diagnosing process anomalies[2,3,4]. The most significant advances in the application of expert systems to process diagnostics has involved construction of computer models with interconnecting nodes. General principles of the plant process are used to attempt to isolate the cause of a possible anomaly to one of the nodes. Another approach to process diagnosis is to identify the cause of a possible anomaly with an event-cause model of the process[5,6].

The related concept of using expert systems for detecting possible anomalies in plant instrumentation has received little attention. In the works previously mentioned, the methods of anomaly detection were the standard methods of signal validation which rely on detection of statistical deviations in sensor outputs from normal values.

This paper describes a method of signal validation using expert system technology which detects possible anomalies in a sensor's output much as an operator does.

The operator looks for possible anomalies in a sensor's output by comparing output characteristics of a sensor, such as noise level and average output, to that of its redundant sensors, if any are available, and checking for output characteristics that he considers normal for the sensor. The system can be used to quickly scan over an array of sensor outputs and flag those which are observed to have possible anomalies. This system implemented in an operating process plant could be used for continuous, on-line anomaly detection with a minimum of computational effort.

1.2 Statement of Problem

A menu driven expert system has been developed that provides information about the sensors used to monitor nuclear power plant subsystems and performs automated signal validation on those sensors. The program displays the names of sensors used to monitor a plant subsystem and then answers the user's questions concerning the sensor's location and type, current output characteristics or expected output characteristics. Upon the user's request the expert system checks for possible anomalies in a specified sensor's output. The evaluation is designed to mimic the operator's ability to detect anomalies in sensors

by comparing the sensor's observed behavior to the sensor's expected behavior. The program automatically performs the evaluation by using comparisons with redundant sensors, when available, and knowledge about the operating characteristics of the system being monitored to assist in the evaluation.

1.3 Purpose of the Study

Initial interest in the work described in this report was a result of a signal validation concept that uses a variety of methods, such as noise analysis, generalized consistency checking and others[7], to perform complete signal validation. The traditional methods of signal validation rely heavily on numerical calculation to perform the analysis. The approach developed in this paper performs signal validation as an operator would, using limited calculations.

A data base was obtained that contains information on all available sensor failures that led to a power reduction or shutdown in nuclear power plants in the United States. The data base listed the type of sensor that failed along with a brief description of how the sensor failed and how the failure was detected. The majority of sensor anomalies that were detected by the plant operators were.

1. The average sensor output was not what the operator expected for current operating conditions of the plant.

2. The sensor failed to respond to a change in the operating condition of the plant.
3. The sensor's output failed to fluctuate with normal process noise.

These possible anomalies were usually detected in one of two ways.

1. The sensor output was in disagreement with its redundant sensors.
2. The operator observed sensor output that he did not expect.

The original concept of this portion of the project was to mimic an operator's ability to perform signal validation with minimal computational effort. The concept was to use comparisons with redundant sensors to perform signal validation whenever those sensors were available, and to complement the results of the comparison among redundants by using knowledge about the system being monitored to predict expected signal behavior and then compare that to observed behavior.

All of the procedures required to perform the evaluation are hidden from the user and the final result was originally the only answer returned. The idea of an automated validation system has been expanded to allow a user access to any information that is produced as part of the evaluation and any reasoning used to reach the final conclusion of the evaluation. This allows the system to be

a stand-alone information package that is automatic and generic to any group of sensors represented in the proper format.

The program was developed primarily in LISP with two external routines defined in C. The external routines perform calculations on digitized sensor output stored in data files.

1.4 Assumptions and Limitations

The program was developed with only two assumptions. The first assumption is that the sensor's output prior to the consultation is available in a properly formatted data file. The second assumption is that the sensor's output was acquired during steady-state operation. The signal validation portion of the program evaluates steady-state conditions or changes to and from steady-state conditions and is not concerned with the path taken to any current operating condition.

CHAPTER II

GENERAL DISCUSSION OF THE EXPERT SYSTEM

2.1 Introduction

The basic components of an expert system are the knowledge base, the inference engine, and the input/output interface. The knowledge base of this expert system is a representation of the sensors used to monitor a subsystem of a power plant. The inference engine consists of the routines used to extract information from the knowledge base. The input/output interface is the means by which an expert system communicates with the user to ask questions, receive answers, and display results.

2.2 The Knowledge Base

The knowledge base of the expert system is a list of the sensors used to monitor a subsystem. An internal structure in the LISP environment, called SENSORLOOP, has been defined for this purpose. The variables in the SENSORLOOP structure are

1. name,
2. location,
3. type,
4. current-reading,
5. previous-reading,
6. fluctuation,

7. minimum-fluctuation,
8. maximum-fluctuation,
9. limit0,
10. upper-limit1,
11. lower-limit1,
12. upper-limit2,
13. lower-limit2,
14. sensitivity,
15. sensitive-to,
16. error,
17. filename,
18. result-filename.

The name, location, and type of a sensor are obtained from the piping and instrumentation diagrams of the system that is being monitored. The current-reading is the average of a sample of the sensor's output, which has previously been recorded and stored in the data file, filename. The previous-reading is the current-reading at the time of the previous evaluation. The fluctuation is the standard deviation about the absolute value of the mean of the sensor's current average output. The minimum-fluctuation and maximum-fluctuation are the expected upper and lower limits for normal fluctuations. The variable limit0 represents the power beyond which the normal limits of operation are set by upper-limit2 and lower-limit2. At power levels below limit0, upper-limit1 and lower-limit1

are used to set the limits. These limits are the range of expected sensor output for current operating power under normal control strategy. The sensitivity is the value of expected change in a sensor's output for an observed change in an independent variable, sensitive-to. The error of a sensor is the overall expected accuracy of the sensor loop as calculated by instrumentation engineers. The result-filename is the data file used to record the results and explanation of the results produced by the signal validation portion of the software package.

All sensors are represented by the SENSORLOOP structure and are maintained in the list SENSORLOOPS.

2.3 The Inference Engine

The inference engine of this system is generic to any list of sensors that is maintained with the sensorloop structure. The inference engine consists of a set of routines used to provide a user with information concerning the location and type of a sensor, sensors that are redundant to other sensors, current output and expected output of a sensor and an algorithm to chain through a set of if/then or production rules to perform signal validation on a specified sensor. Production rules have the following form

```
If (assertion 1 is true) and
   (assertion 2 is true) and ...
   (assertion n is true)
```

THEN

(consequence 1 is true) and

(consequence 2 is true) and ...

(consequence n is true).

The chaining algorithm uses a set of routines to reach a final conclusion about the performance of a selected sensor by attempting to prove assertions that will lead to the verification of a sensor anomaly. The assertions which are used to determine if these anomalies exist are mathematical expressions that evaluate to a true or false answer, or they are string comparisons that also result in true or false answers. The inference engine contains a routine that evaluates the assertion and returns the result of that evaluation.

The chaining portion of the inference engine chains through a set of rules which contain the logic required to prove that a sensor has a possible anomaly. The chaining algorithm is an extension of an algorithm developed by Winston and Horn, in their book, LISP [8].

The other routines of the inference engine are used to determine information about a sensor and its relationship to other sensors, or the subsystem it is monitoring.

2.4 The Input/Output Interface

The input/output interface of this system performs multiple tasks. This portion of the expert system is used to display all available options to the user, to obtain requests for information from the user, and to display the response to the user's request.

The display of available options is done with a menu screen. The user selects options by number and the requested information is displayed in a natural language format.

The results of the signal validation portion of the system are shown using two features. The first feature allows the user to follow along through the evaluation by listing the rules that were used to reach the final conclusion about a sensor's performance. The second feature is a written explanation of how the system reached its final conclusion. The summary tells which tests passed or failed, why they passed or failed, and how their results led to the final conclusion.

The expert system was developed primarily in LISP, using routines written in C to perform any extensive calculations required. LISP involves symbolic representations of objects and relationships between objects. AI languages, such as LISP, lend themselves to expert system applications due to their ability to perform symbolic manipulation. They are usually not well-suited to

perform large numerical calculations as compared to the FORTRAN and C languages [9].

The chaining algorithm developed by Winston and Horn has been modified to enable the use of confidence factors to represent the confidence of a conclusion based on the results of the tests used to detect possible sensor anomalies.

CHAPTER III

INFORMATION FLOW

3.1 Updating Sensor Readings

All the variables used to describe the current condition of the plant or the sensors are first updated when a consultation begins. Data files, external to the LISP environment, are used to store a digitized sample of the current outputs of the sensors, and the value of the current power level of the plant. The current power level is the variable typically used to define the operating state of a nuclear power plant. Many plant parameters, such as steam generator level and feed water flow, are controlled from the power level; other noncontrolled parameters, such as hot leg temperature and steam dome pressure, are functions of power level. Since LISP is not specifically suited to perform extensive calculations, the updating calculations are performed by routines written in C and called externally from LISP.

The value of the current power level of the system, the current-reading and fluctuation for all sensor loops are updated with call-out routines. A call-out routine is used to execute functions external to the LISP environment. A call statement is used to pass variables from LISP to a function stored externally. The function is executed and a

final value is returned to the LISP environment. The value of any variable can be set to equal the returned value.

The value of current-power, the variable used to represent the percentage of rated thermal power, is updated with the following call statement:

(setq current-power (call-out average "power.dat"));

where average is the name of the external routine that averages the data in the file named "power.dat".

The LISP routine UPDATE is used to update the SENSORLOOP variables current-reading and fluctuation. UPDATE takes as input the name of the sensor whose values are to be updated. The LISP routine sends the name of the filename, stored in the SENSORLOOP structure, to the C routine called AVERAGE, which opens the file, averages a sample of data points, and returns the average of those points to the LISP environment. The returned value is set to be the value of the variable current-reading. Next the UPDATE routine updates the value of the fluctuation value.

Another C routine has been written, called FLUCTUATION, that is called similarly to the routine AVERAGE, but returns the standard deviation about the mean of the readings in the data file.

3.2 Task Selection

The top level of the expert system displays a list of names of the sensors that are represented in the current sensorloop list, and a list of information that can be

obtained about those sensors. There are two types of information that the expert system can report to the user. The expert system can give general information about the arrangement of sensors in the subsystem or about the current characteristics of a specified sensor.

3.2.1 Sensor Information

There are two options for general information about the sensors that are monitoring the system.

1. Obtain a list of all sensors of a specific type.
2. Obtain a list of all sensors at a specific location.

The sensor specific tasks are

1. list characteristics of a sensor,
2. obtain a list of sensors redundant to any other,
3. obtain an average of the redundant sensor's outputs,
4. obtain a list of all sensors at the same location as any other,
5. obtain a list of all sensors in the subsystem that are the same type as any other,
6. perform signal validation of a sensor.

The general information tasks are used to give an operator information about which sensors are monitoring the system. One task presents a list of the possible locations of sensors for this list and then allows the user to select one of these locations. All of the sensors at the

specified location are then listed for the user. The other task does the same for the type of sensors used to monitor the subsystem.

Whenever a sensor specific test is selected the system prompts the user to enter the name of the sensor to be examined. The input string is then compared to the sensors in the sensorloop list and if a sensor with the input name does not exist, the user is shown the list of sensors again and allowed to pick another.

Upon selection of a legitimate sensor, the system determines the characteristics of the sensor that is available from the knowledge base of sensorloop representations. Lists of sensors which have certain relationships to the specified sensor are created by comparisons among the variables stored in the SENSORLOOP structure. The lists which are currently being created are

1. same-location,
2. same-type,
3. redundant,
4. upstream,
5. and downstream.

The locations of a subsystem are divided into regions where the properties being measured are not expected to differ significantly. Sensors common to the region of the selected sensor have the same location variable. The list same-type is a collection of all sensors in the subsystem

that measure the same property as the selected sensor. The list of redundant sensors is the intersection of the same-type and same-location list.

Creation of the lists of sensors which are located upstream and downstream from the location of a specified sensor must be handled differently. In order to make these lists the expert system must have a representation of the configuration of the subsystem. This representation is achieved by assigning an integer value to a sensor location and giving the location of the region directly upstream as the value of that integer minus one and the region directly downstream as the value of that integer plus one. The upstream and downstream lists are created by finding sensors whose location value minus the selected sensor's location value evaluates to plus or minus one, respectively.

One possible selection allows the user to obtain the characteristics of the selected sensor. The available characteristics are

1. sensor's current average output,
2. sensor's average output at the time of the previous evaluation,
3. the change in output since the previous evaluation,
4. the upper and lower limits of normal sensor output at current power,
5. the limits of expected fluctuation of the

sensor's output,

6. the current fluctuation of a sensor's output,
7. the sensor's location,
8. the type of sensor,
9. the physical units of the sensor output.

The user is allowed to select as many of these options as needed. When the operator has obtained the desired information, he enters a "q" for quit and is then returned to the top level, where he has the option of ending the consultation or selecting another task.

Other sensor specific tasks a user can request are

1. listing of any sensors which are redundant to the selected sensor,
2. calculation of the average of the current readings of the redundant sensors,
3. listing of the sensors at the same location as the selected sensor,
4. listing of the sensors measuring the same parameter as the selected sensor,
5. and evaluating a sensor's condition.

The last option available is for the system to perform signal validation on the specified sensor.

3.2.2 Sensor Evaluation

From the evidence compiled concerning sensor failures it has been observed that operators usually detect sensor failure in one or more of the following ways.

1. Sensor output is not normal for current operating conditions.
2. Sensor responded incorrectly to a change in the operating condition of the system being monitored.
3. Sensor output ceased to fluctuate with normal process noise.

Whenever there are redundant measurements available, an operator does not need specific knowledge of the system in order to perform signal validation. Usually an accurate assessment of a sensor's behavior can be determined by a comparison of its output against that of the redundant sensors. There are many situations in which redundant sensors are not available for the analysis, or the comparison among the redundant sensors is inconclusive, as in the case of two redundant sensors disagreeing with each other. When a sensor with inconclusive redundant information has become suspicious, and an operator must decide whether it has failed or not, he can only compare the sensor's output and characteristics to those he expects. He must rely on his knowledge about the system to provide the information needed for the comparison.

The signal validation portion of the expert system is designed to detect anomalies in a sensor's output that may be indicative of a possible sensor failure the same way as an operator would. The result of the evaluation includes confidence factors from zero to one hundred percent, that a sensor has the three anomalies most commonly detected by operators.

Each possible anomalous condition is tested for using two procedures. The first test is a comparison of the sensor's characteristics against the same characteristics of redundant sensors, if any redundant sensors are available. The second test incorporates knowledge about the system being monitored to predict expected sensor behavior and then compares that to observed sensor behavior. Results from the two tests are used to determine the confidence factors that the anomalies exist. The tests that are used to complement the comparison among redundant sensors are

1. limit checking for normal limits of sensor output at current power level,
2. sensitivity analysis of a change in a sensor's output for a change in the operating conditions of the system,
3. and a limit checking for normal noise levels in a sensor's output.

The comparison of a sensor's average output with the outputs of the redundant sensors is complemented by a limit checking scheme. The comparison of a sensor's change in average output from the previous evaluation is complemented by a sensitivity analysis. The comparison of the noise level in a sensor's output is complemented by a comparison of the current noise level with the limits of expected noise level. The information required for the system specific tests are stored in the SENSORLOOP structure for each sensor.

Whenever there are no redundant sensors or the analysis of the redundant sensors proves inconclusive, the complementary evaluations are the sole basis for the final conclusions of the evaluation.

3.2.3 Comparison with Redundant Sensors

The most useful sensor for assisting an analysis is probably a redundant sensor. Under normal operating conditions redundant sensors should behave similarly. The first group of sensors that the expert system attempts to utilize is therefore the redundant group. A comparison against redundant sensors is performed on three variables in the SENSORLOOP structure.

1. Current-reading,
2. change, and
3. fluctuation.

A decrease in the noise level maybe a symptom of a sensor with a clogged sensing line or some other malfunction. Often a sensor will continue to output a constant value but will not respond to rapid system changes. If there have been no changes which should have caused the sensor's output to change, the failure can easily go undetected. Checking the noise level is one way of determining whether a sensor is responding to normal process fluctuations.

The redundant test assumes a maximum of five redundant sensors and will come to one of the following conclusions based on the number of redundant sensors, the agreement among the redundant sensors and their agreement with the sensor being evaluated.

1. Strongly conclusive that the sensor is normal.
2. Fairly conclusive that the sensor is normal.
3. Slightly conclusive that the sensor is normal.
4. Inconclusive.
5. Slightly conclusive that the sensor is anomalous.
6. Fairly conclusive that the sensor is anomalous.
7. Strongly conclusive that the sensor is anomalous.

There are three values calculated for each redundant analysis. Red-agree represents the ratio of the number of sensors in the redundant list whose specified variable is within specified tolerances of the sensors being evaluated, to the total number of sensors in the redundant list. Red-

groups represents the number of different values, within tolerances, for the specified variable observed among the redundant sensors. The third value necessary for the redundant evaluation, length is simply the number of redundant sensors. Length is the LISP function used to determine the number of objects in a list. This function is used to determine the number of sensors in a list of redundant sensors.

The procedure used to reach one of the seven possible conclusions for the redundant analysis are shown in Figs. 1-7. The conclusion written at the top of the figure is the result of the comparison among the redundant sensors. This is true if the values of the length of the redundant sensor, the number of groups of sensors with similar output characteristics within the redundant list, and the number of sensors with similar output characteristics agreeing with the sensor being evaluated are such as to allow the diagram to reach a terminal node.

For example, during an evaluation the expert system will attempt to determine if the comparison among redundants is fairly conclusive that the sensor is anomalous. Figure 6 is a diagram illustrating the conditions under which this conclusion is true. This logic pattern is programmed into the rules of the system. From the diagram it can be seen that there are four initial paths that the system can evaluate to prove this conclusion. If the number of redundant sensors is five the

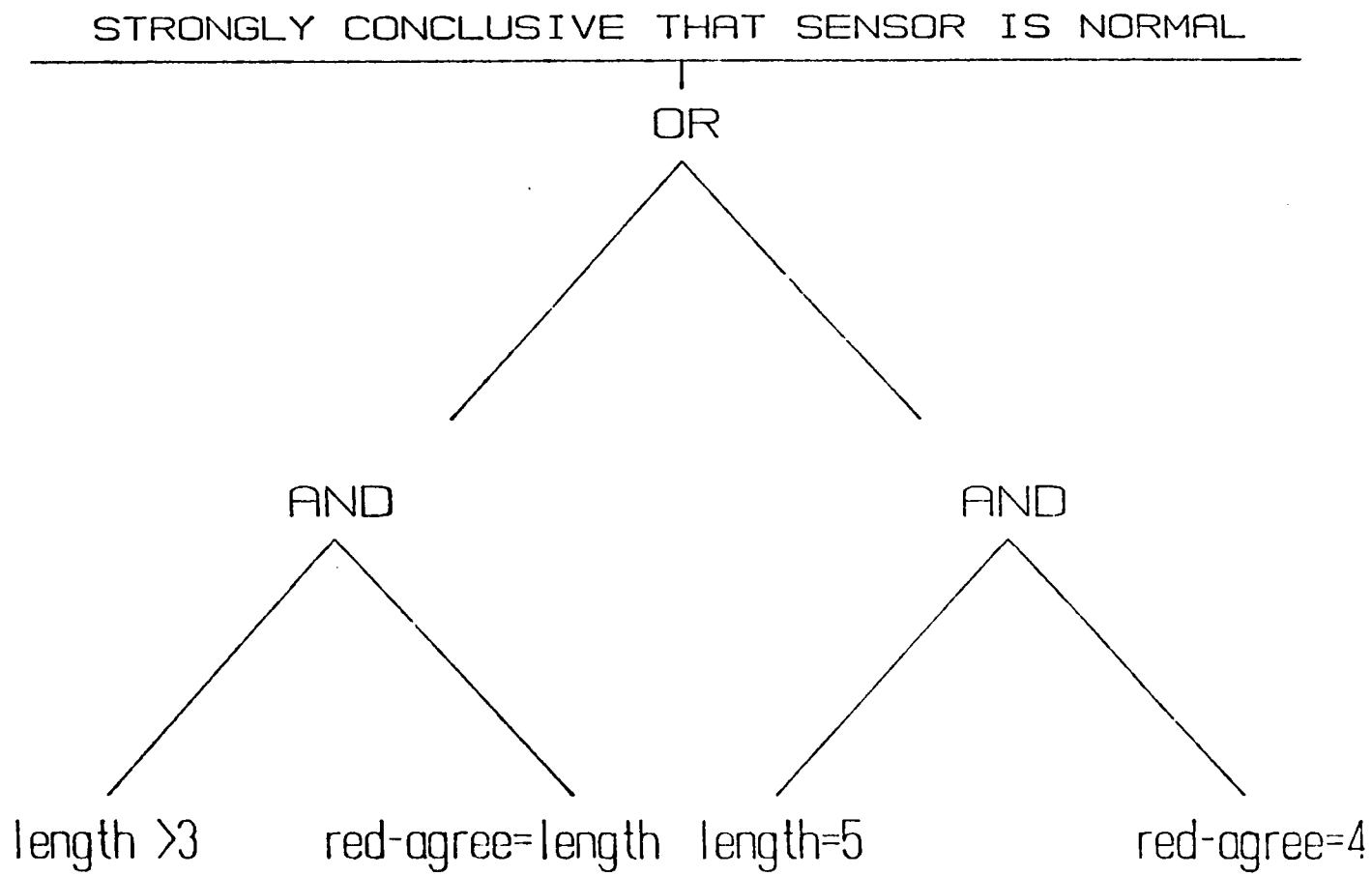

Figure 1. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion one.

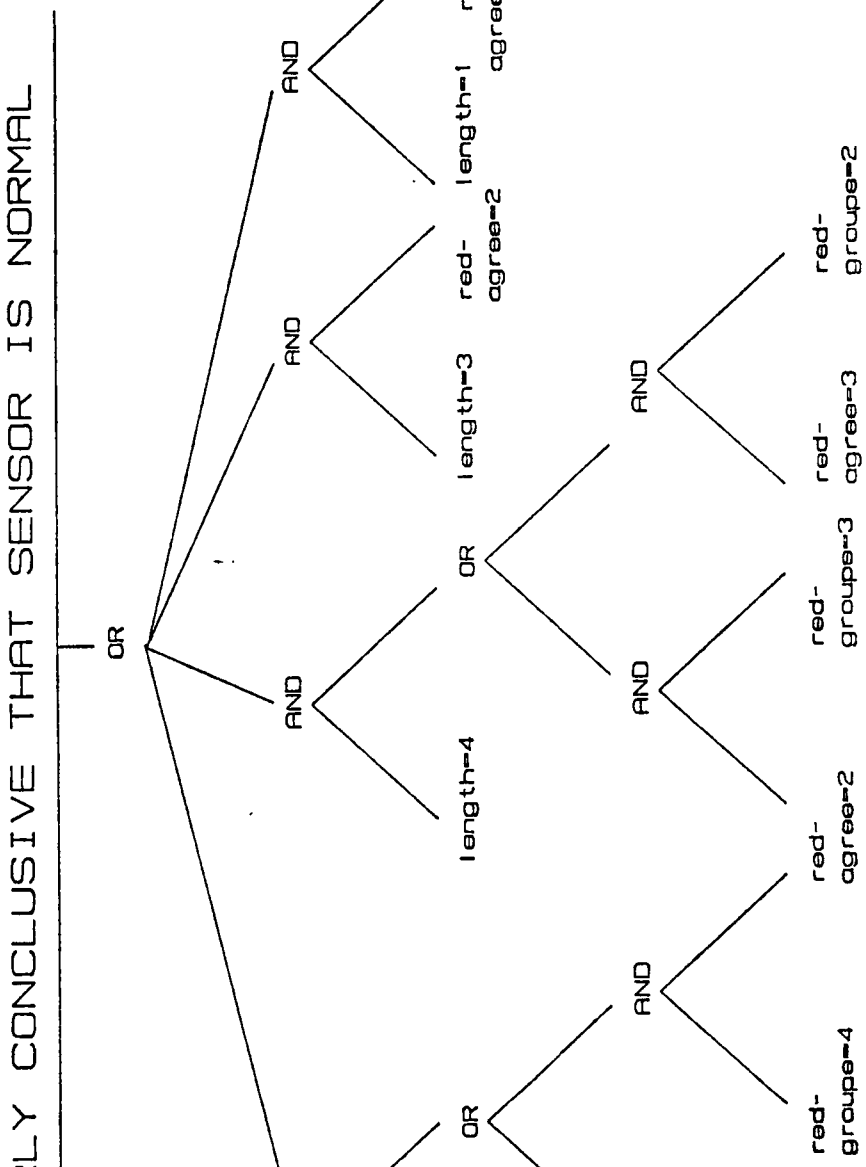

Figure 2. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion two.

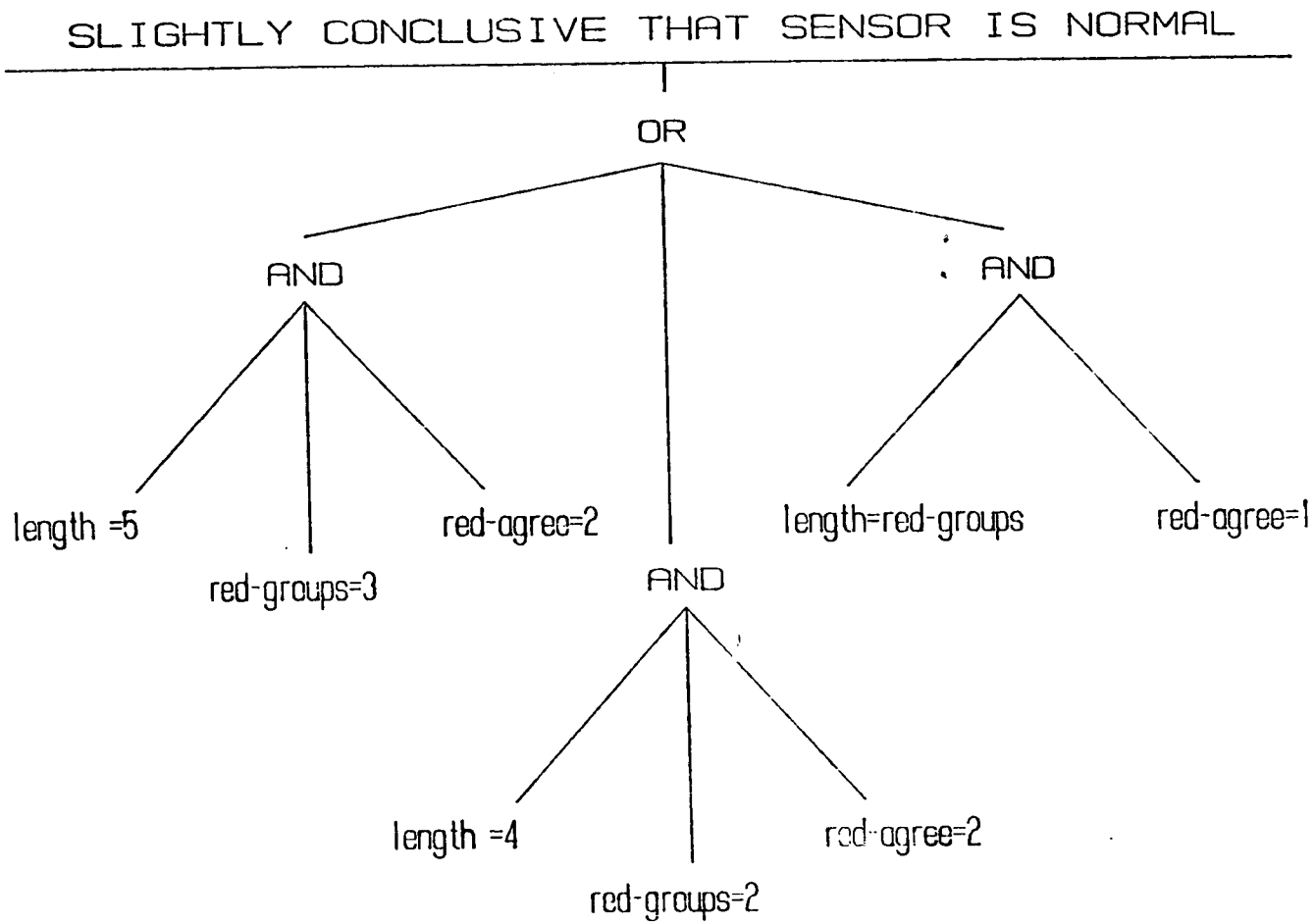

Figure 3. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion three.

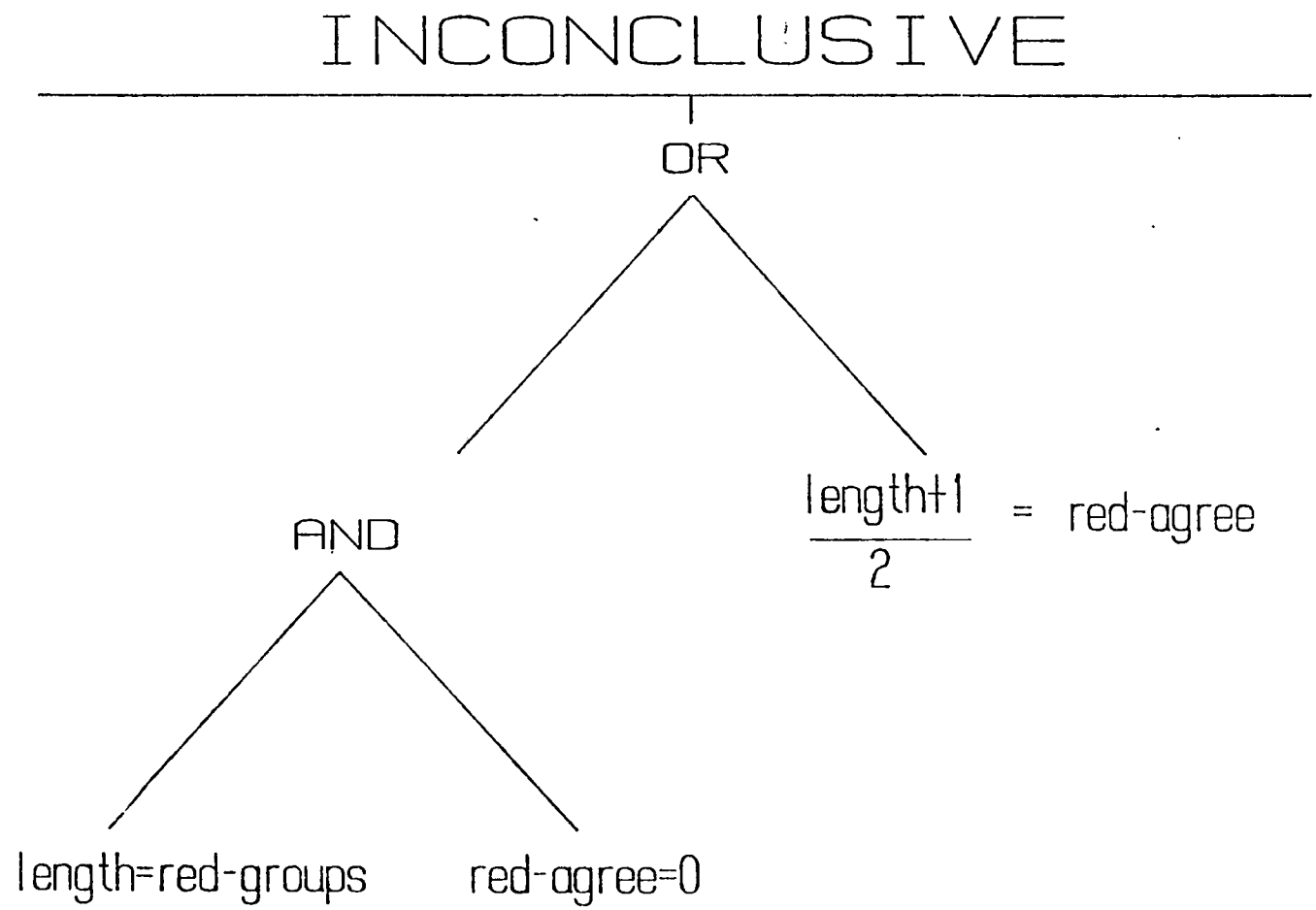

Figure 4. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion four.

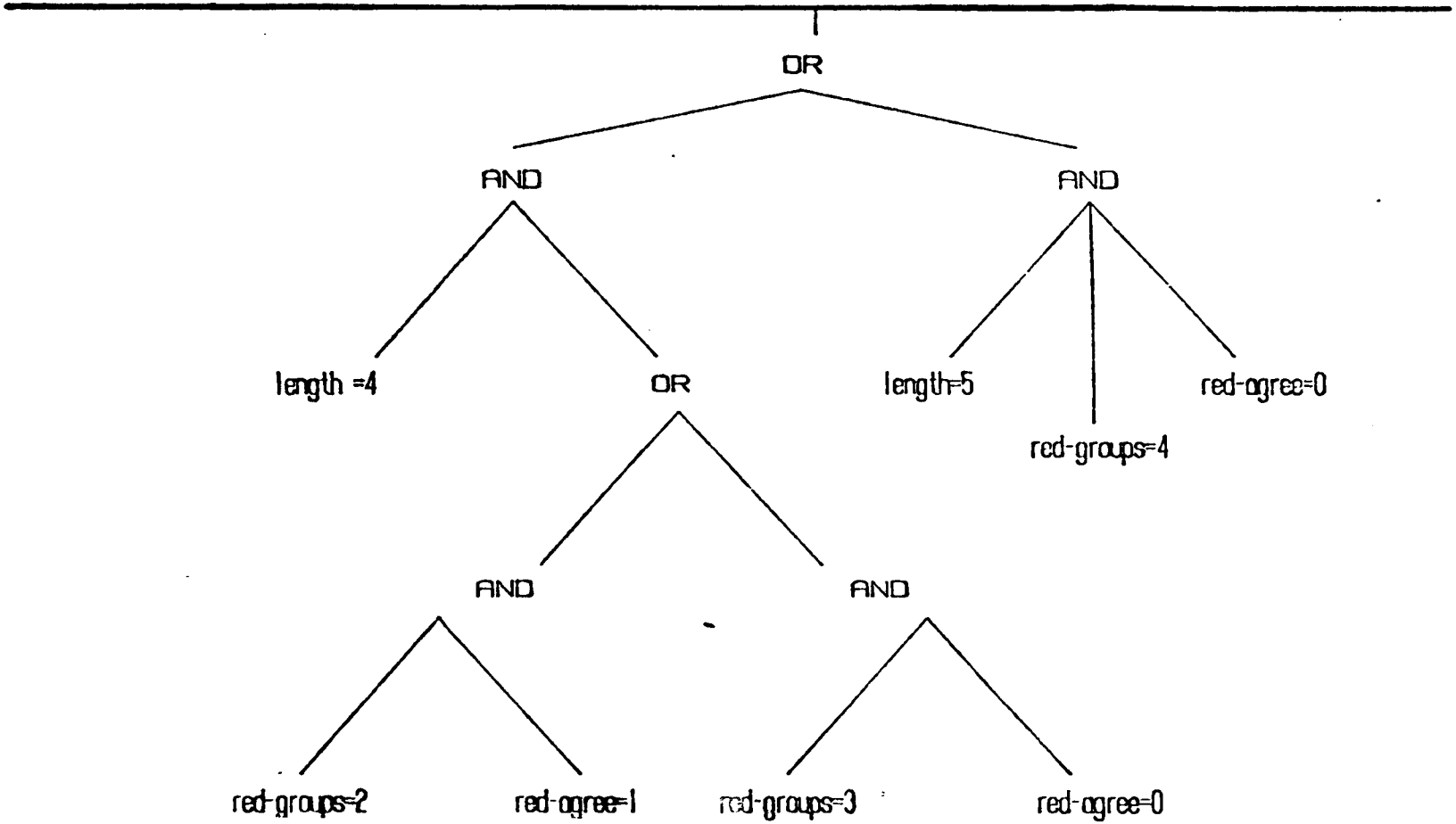

Figure 5. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion five.

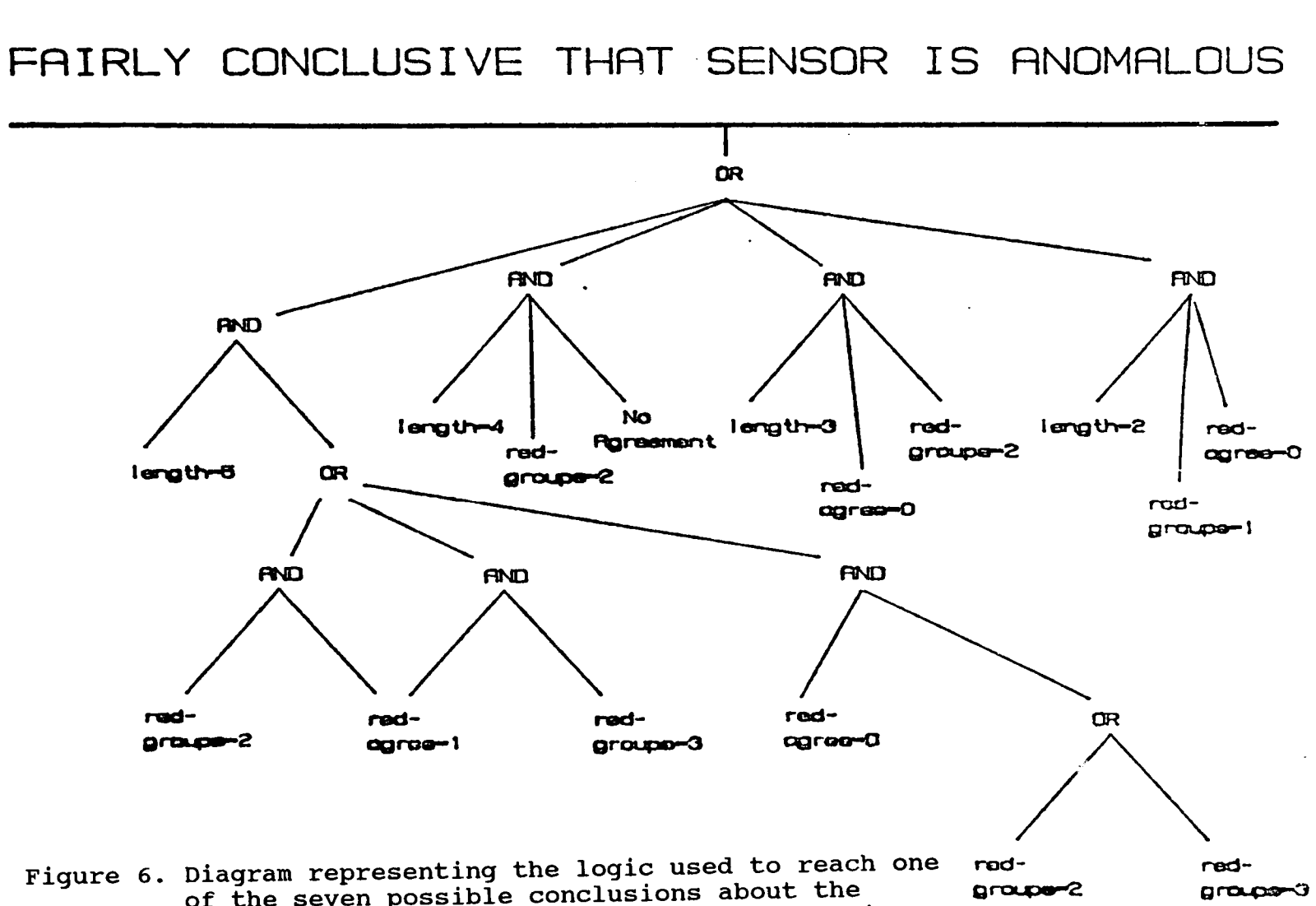

Figure 6. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion six.

STRONGLY CONCLUSIVE THAT SENSOR IS ANOMALOUS

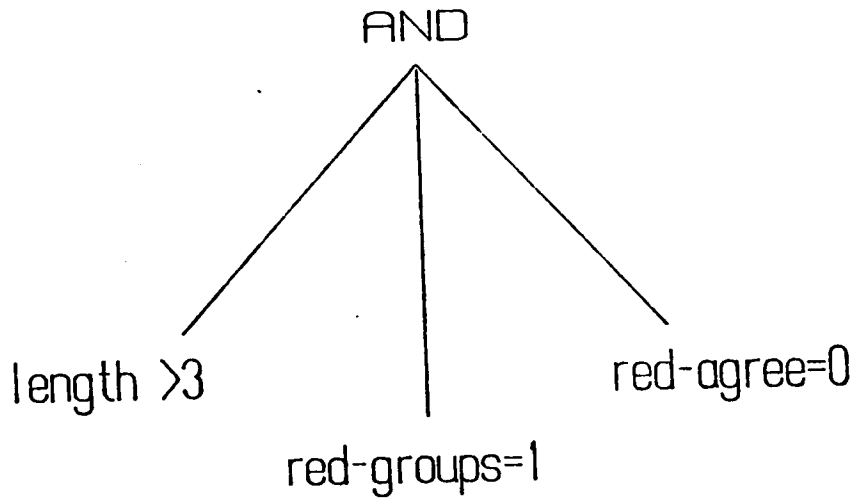

Figure 7. Diagram representing the logic used to reach one of the seven possible conclusions about the comparison among redundant sensors, conclusion seven.

path to the far left must be evaluated. Following that logic pattern, there are three possible paths that can be proven true in order for the conclusion to be true. Again, following the extreme left path, if the number of groups within the list of redundant sensors with similar output characteristics is two, then the number of sensors in the redundant list with output characteristics similar to the sensor being evaluated must be one in order for the conclusion of the comparison among redundant sensors to be fairly conclusive that the sensor is anomalous.

3.2.4 Analysis Using System Specific Information

Redundant sensors provide a wealth of information for performing signal validation. Unfortunately redundancy is not always available and other methods of validation must be used. Operators also detect failed sensors by observing deviations between expected sensor output and observed sensor output. To mimic an operator's ability to detect failures in this manner three evaluations based on system specific information are used. The first test checks to see if the change in sensor output is what is expected due to a change in the operating condition of the system, the second checks for sensor outputs that are within expected limits for the current power level of the system, and the third checks for proper noise level in a signal.

3.2.4.1 The Sensitivity Analysis

The sensitivity analysis is used to predict the direction and approximate magnitude of changes in the sensor output for changes in an independent parameter. The sensitivity coefficient is a number relating the expected sensor output change for a change in a specified parameter under normal control operations. A sensor may be sensitive to

1. no other variable,
2. power, or
3. the output of any other sensor represented in the sensorloop list.

The parameter that a sensor is sensitive to and the sensitivity to that parameter must be determined from plant data. The sensitivity analysis checks for correct magnitude of the output change. The expected magnitude of the sensor output change is predicted from the sensitivity coefficient and the amount of change of the independent parameter, sensitive-to. The predicted magnitude of sensor output change is then compared with the actual magnitude of the change. If the predicted and observed changes in the sensor's output are within the error tolerances of the sensor, the test has passed. The only possible value of this portion of the sensitivity analysis is true or false.

3.2.4.2 The Limit Checking Analysis

Another system specific test is designed to determine whether a sensor's output is within normal limits for the current power level of the plant. The limits needed to perform this analysis are stored in the SENSORLOOP structure for each sensor. Limit0 is the parameter that is used to determine which limits the program is supposed to associate with the sensor output for the current operation. If the power is below limit0, the limits to be used are upper-limit1 and lower-limit1; otherwise upper-limit2 and lower-limit2 are used. The system can easily be modified to handle as many limits as are required to describe the entire range of sensor operation. The limit checking analysis is passed if the current sensor output lies between the limits specified by the limit variables. Again the limit checking test has only true or false for possible answers.

3.2.4.3 The Test for Proper Noise Level

The final system specific test checks for proper noise level in a sensor's output. All normal sensors will have some fluctuation of the output about an average value. A decrease or an increase in the noise level of a sensor's output may be a symptom of sensor failure. Often a sensor will continue to output a constant value but will not respond to rapid system changes. If there have been no

changes which should have caused the sensor's output to change, the failure can easily go undetected. Checking the noise level is one way of determining whether a sensor is responding to normal process fluctuations.

From a sample of real data a minimum and a maximum expected noise level of each sensor is determined. The values of the expected noise level are set by averaging the deviation of a sensor's output about the mean for samples of steady-state data. Ideally the samples should correspond to steady state conditions for every ten percent power level increase between zero and one hundred percent power. The upper limit of normal noise level for a sensor is three times the average of the noise levels in the sample files. The lower limit of the normal noise level is one third of the average value. The use of this range of limits is arbitrary.

The calculated value of the limits of expected standard deviation about the mean of a sensor's output is then stored in the SENSORLOOP structure. If the current output of the sensor has a standard deviation which is less than that of the minimum-fluctuation or greater than that of the maximum-fluctuation the test will fail. The only possible results of this analysis are true or false.

3.2.4.4 Determination of Final Conclusions

For each evaluation, the six tests are run and conclusions about the operating condition of the sensor are

determined using results of these tests. The final output of the expert system is a confidence factor, between zero and 100 percent, for these three possible conclusions.

1. Sensor is possibly anomalous due to incorrect output.
2. Sensor is possibly anomalous due to incorrect response to a system change.
3. Sensor is possibly anomalous due to incorrect response to process noise.

The confidence factor for the possible conclusions are a result of the comparison of the appropriate characteristics among the redundant sensors and the system specific test designed to detect that specific anomaly. If there are no redundant sensors available for the analysis the final result will be dependent only on the accuracy of the system specific information used to predict sensor behavior.

The logic pattern for the final analysis is strongly dependent on the results of the comparison among redundant sensors. If the comparison among redundant sensors is strongly conclusive that the sensor is anomalous, or that the sensor is operational, the confidence factor for this mode of operation will be zero or 100 percent, respectively. Because it takes strong agreement or disagreement with the redundant sensors for the comparison to be strongly conclusive, the redundant analysis is allowed to overrule the system specific tests in this

situation. Whenever the redundant analysis is less than strongly conclusive of the operational status of the sensor, the system specific tests are used to complement the comparisons among the redundant sensors. The logic used to reach the confidence factor for the possible conclusions are shown in Fig. 8. There are fourteen possible combinations of the results of the comparison among redundant sensors and the results of the test designed to detect a possible anomaly. These possible combinations were ranked from strongest evidence that an anomaly exists to the weakest. If the comparison among redundant sensors is strongly conclusive that the sensor is normal or has an anomaly, the expert system will return a confidence factor for that possible anomaly of zero or one hundred respectively. The remaining possibilities are given confidence factors throughout the range based on its ranking.

There is a difference between an inconclusive comparison among redundants for a sensor that has redundant sensors and one that does not. A comparison with redundant sensors that is inconclusive will result in a ten percent confidence that the sensor has the anomaly. This allows the expert system to flag sensor problems that is not certain of, but it does detect a possible problem. This logic is programmed into the rules of the expert system.

The rules used to evaluate a sensors's performance are composed of two groups. The first portion of the rules

RESULTS OF THE COMPARISON AMONG REDUNDANT SENSORS	RESULTS OF THE SYSTEM SPECIFIC TESTS	CONFIDENCE FACTOR THAT THE ANOMALY EXISTS
STRONGLY CONCLUSIVE SENSOR IS NORMAL	PASSED	0.0
	FAILED	0.0
FAIRLY CONCLUSIVE SENSOR IS NORMAL	PASSED	0.0
	FAILED	0.1
SLIGHTLY CONCLUSIVE SENSOR IS NORMAL	PASSED	0.1
	FAILED	.25
INCONCLUSIVE NO REDUNDANTS	PASSED	0.0
	FAILED	.45
INCONCLUSIVE REDUNDANT SENSORS	PASSED	0.1
	FAILED	.55
SLIGHTLY CONCLUSIVE SENSOR IS ANOMALOUS	PASSED	.55
	FAILED	.75
FAIRLY CONCLUSIVE SENSOR IS ANOMALOUS	PASSED	.75
	FAILED	.85
STRONGLY CONCLUSIVE SENSOR IS ANOMALOUS	PASSED	.99
	FAILED	.99

Figure 8. Determination of the confidence factors of an anomaly based on the results of the comparison among redundant sensors and the system specific test designed to detect the anomaly.

contains the logic scheme used to allow six individual tests to reach a conclusion about the sensor's performance and a second portion contains the logic used to derive the final conclusions about a sensor's performance from the results of the six individual tests.

3.3 Learning Capability of the System

The knowledge of the sensors is maintained in the structure used to represent them. The expert system is allowed to learn by updating the information stored in the sensor representation.

The learning of sensor limits occurs when the comparison against the redundant sensor is strongly conclusive that the sensor is normal and the system specific test indicates that the sensor is possibly anomalous. The system recognizes the discrepancy and overrules the system specific test, thus believing the result of the redundant analysis. To allow the system to know that the current condition of the sensor is to be considered normal operation in the future, the system specific information is updated to include the current parameter.

If the comparison against the redundant sensor's average output is strongly conclusive that a sensor is normal, but the limit test shows the average output to be outside of normal limits, the limits are expanded to include the current average output of the sensor. If the

comparison against the redundant sensor's change in sensor output since the previous evaluation is strongly conclusive that the sensor is normal, but the sensitivity analysis shows that sensor changed incorrectly, the sensitivity coefficient will be recalculated for the value observed during the final change. If the comparison among redundant sensors is strongly conclusive that the sensor has a proper noise level, but the fluctuation evaluation says that the output is outside the normal limits, the fluctuation limits will be expanded to include the current noise level.

3.4 Displaying the Results

During the evaluation, the system displays any intermediate conclusions used to arrive at the final conclusions and the rules used to prove those conclusions. This feature allows the user to follow the logic used by the system as it performs the evaluation.

Upon the termination of the evaluation the final conclusions are printed on the screen if their confidence factor is greater than zero. If results of the conclusions for all three possible anomalies have certainty factors of zero, the evaluation was unable to detect any possible failure and the conclusion of the evaluation will be that the sensor is operational.

After the final conclusions and their confidence values have been presented, the user is given the option of seeing the results of the six tests used to reach that conclusion

and a listing of the facts used to obtain the results of the six tests. These results are presented in natural language format. The results of the evaluation are stored in the sensors result file for further examination by the user.

Upon the user prompt the system returns to the top level mode where the options are presented and the consultation starts again. Figure 9 is a flow diagram showing the logic steps of the expert system. The program is listed in Appendix A.

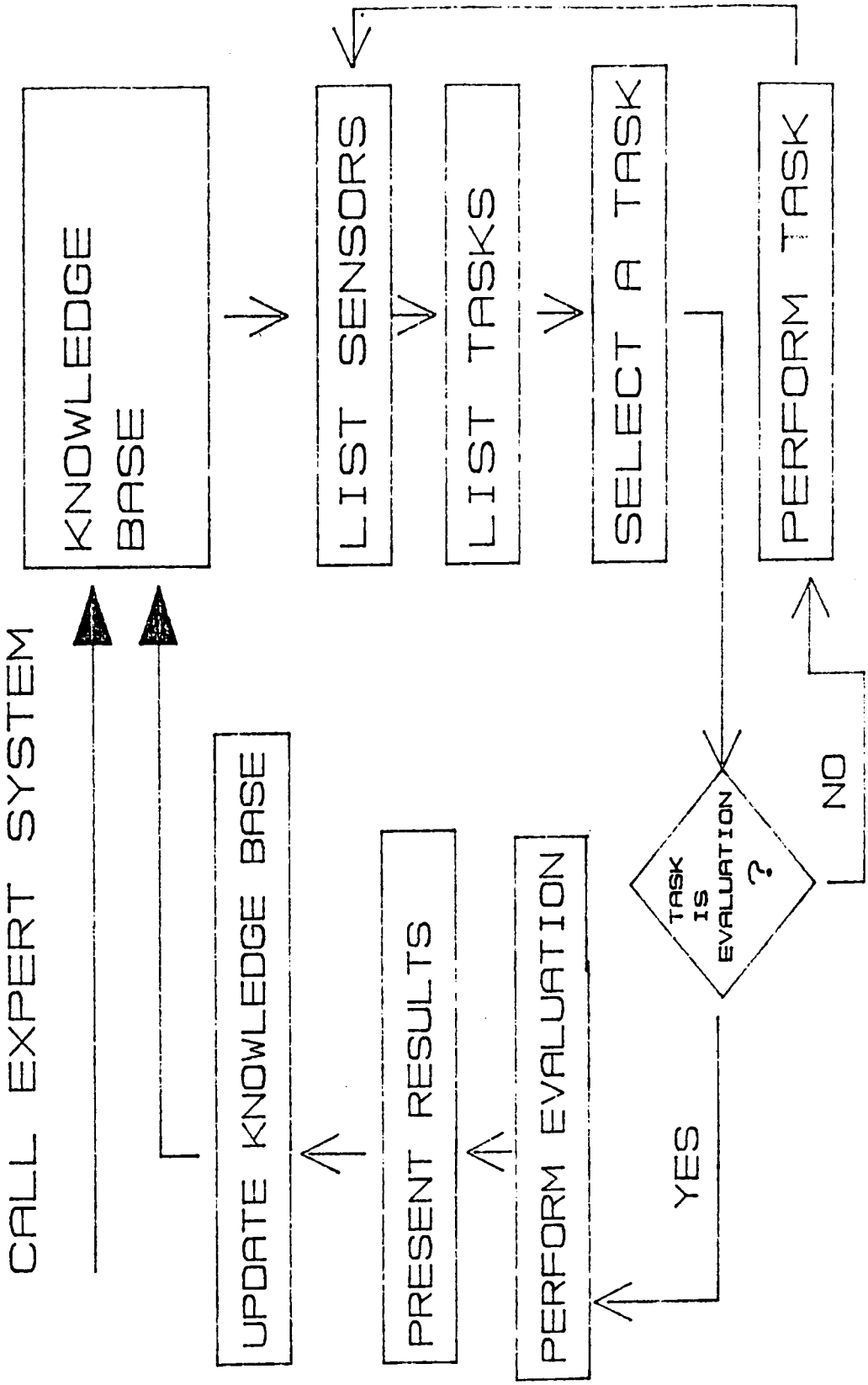

Figure 9. Flow diagram of the Signal Validation Expert System.

CHAPTER IV

APPLICATION TO A PWR STEAM GENERATOR

4.1 Data Acquisition to Build the Knowledge Base

The expert system has been implemented on a steam generator of a Westinghouse four-loop pressurized water reactor for initial testing. Operational data were acquired every minute during the startup of the PWR nuclear power plant for twelve sensors used to monitor a steam generator. The twelve signals recorded were

1. four level measurements,
2. three steam dome pressure measurements,
3. two feedwater inlet flow measurements,
4. one hot leg temperature measurement,
5. one cold leg temperature measurement,
6. one primary coolant flow rate measurement.

The acquired data were used to derive the knowledge base for the operating characteristics of the steam generator. Figures 10 - 15 show the six measured parameters as a function of percent rated thermal power.

Figure 10 is a plot of the steam generator level versus power. The steam generator level is the only controlled parameter among the six parameters. Below 66 percent power it is maintained at a value of approximately 48 feet.

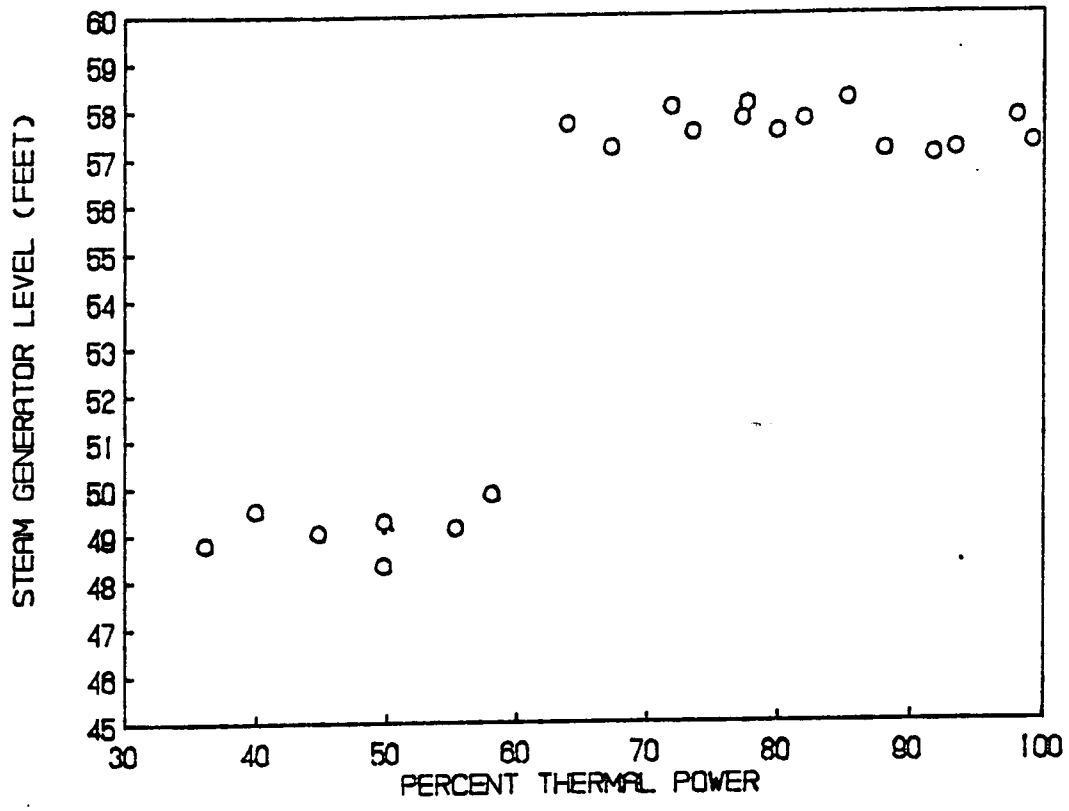

Figure 10. Steam generator level as a function of power for a typical steam generator.

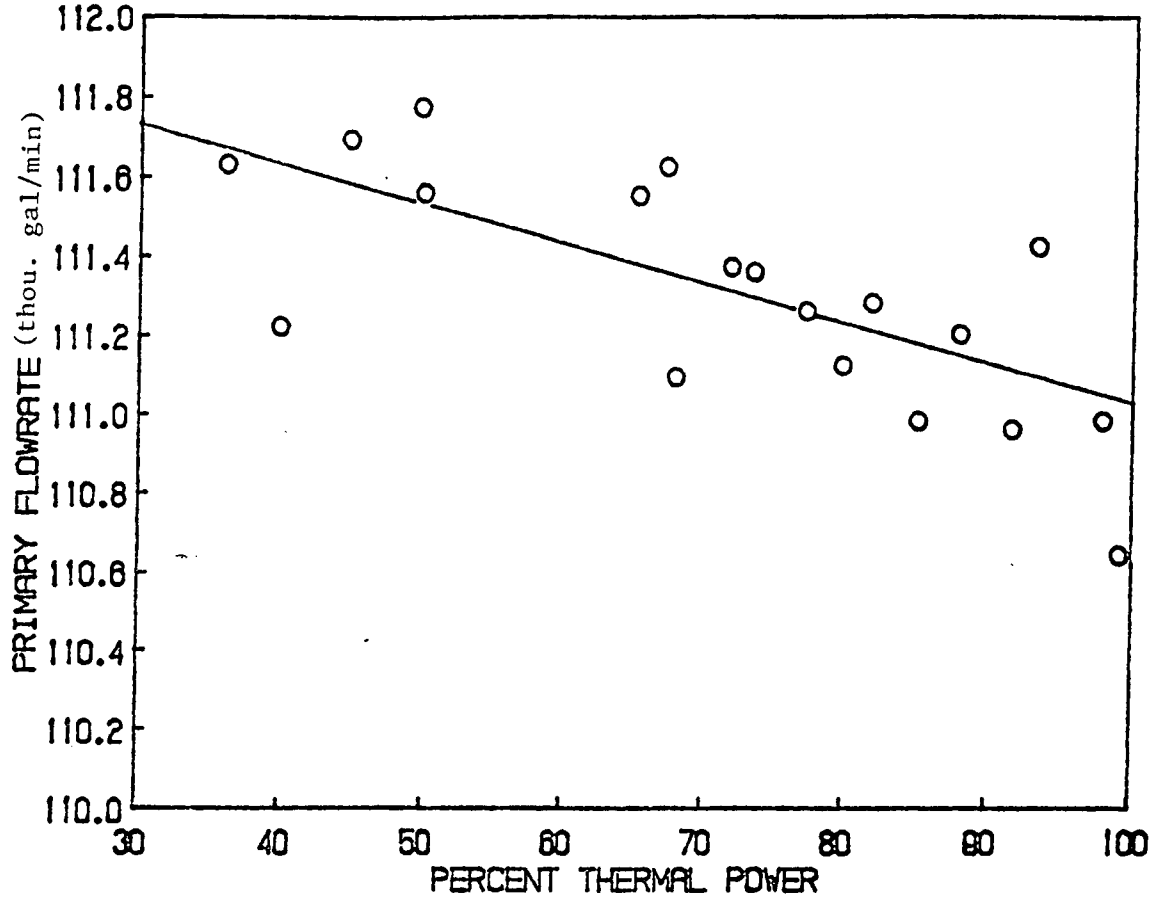

Figure 11. Primary coolant flow rate as a function of power for a typical steam generator.

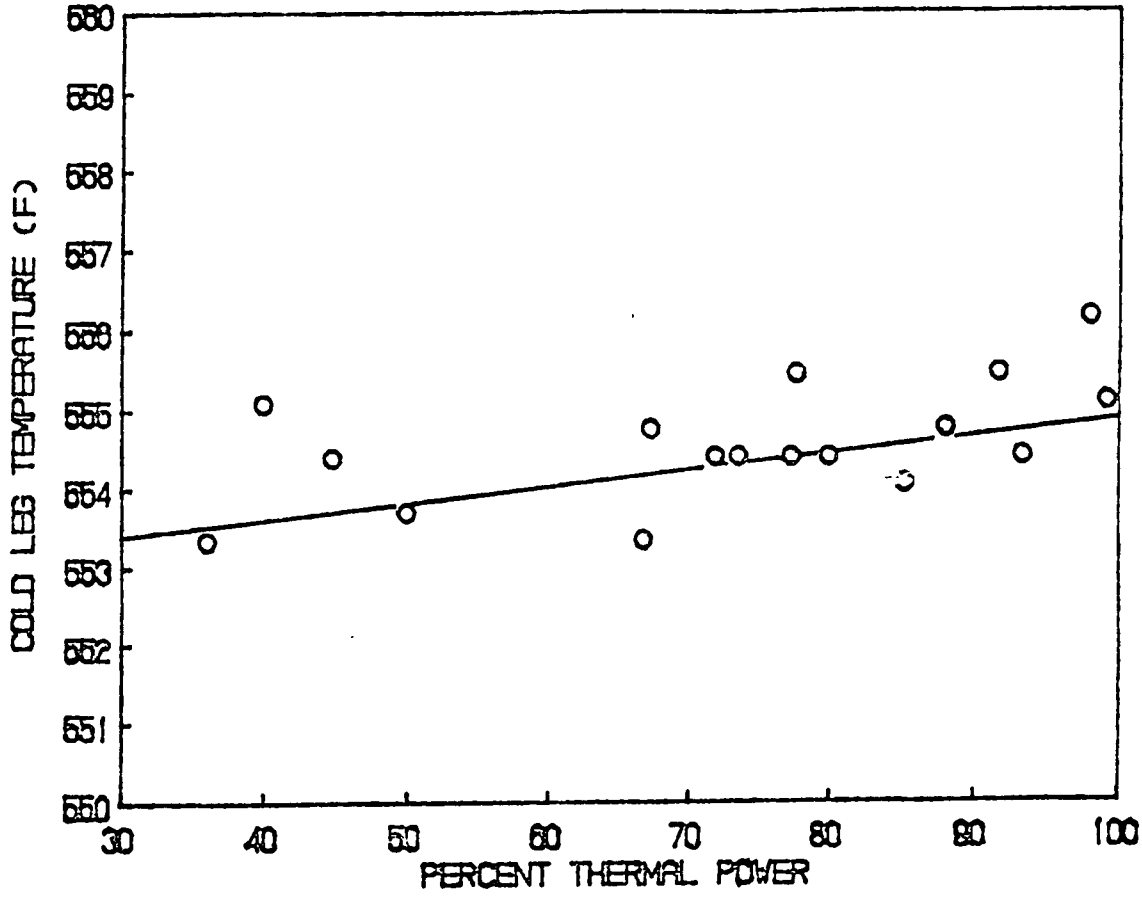

Figure 12. Cold leg temperature as a function of power for a typical steam generator.

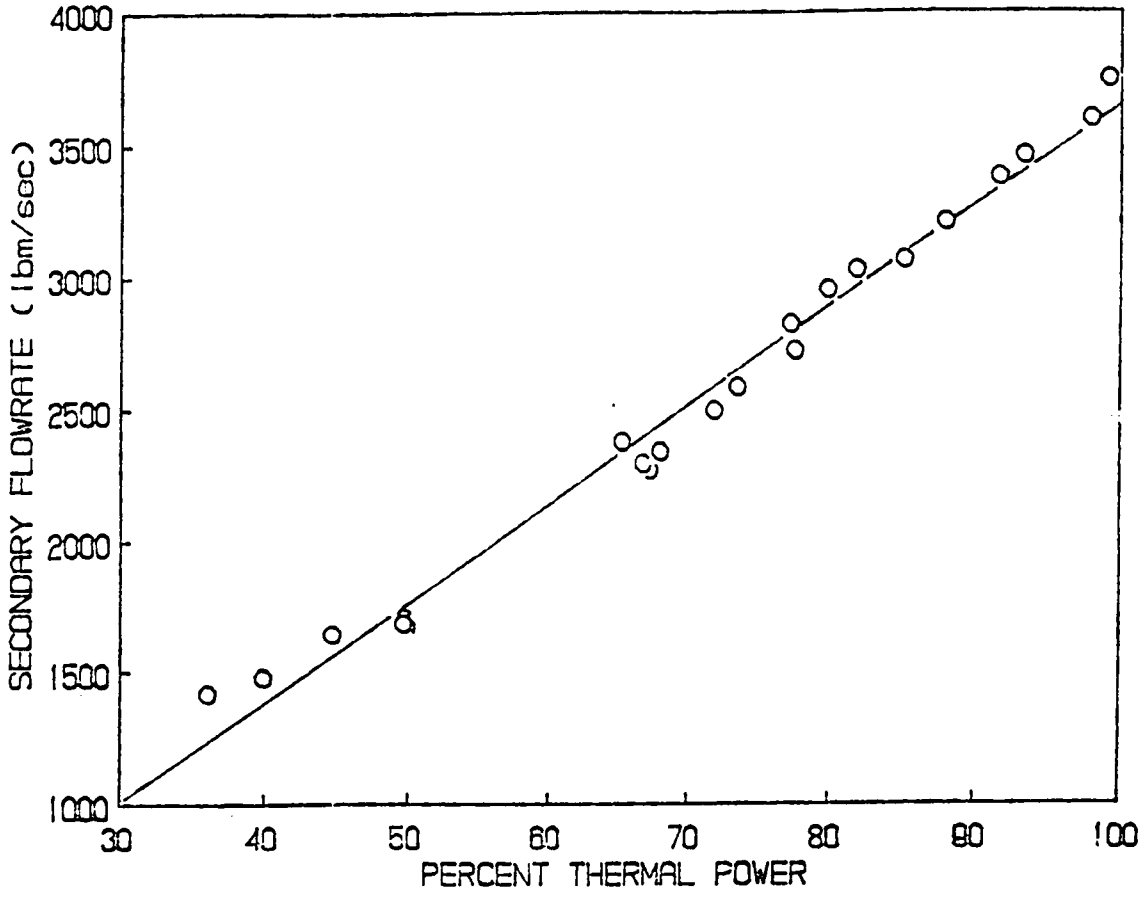

Figure 13. Secondary coolant flow rate as a function of power for a typical steam generator.

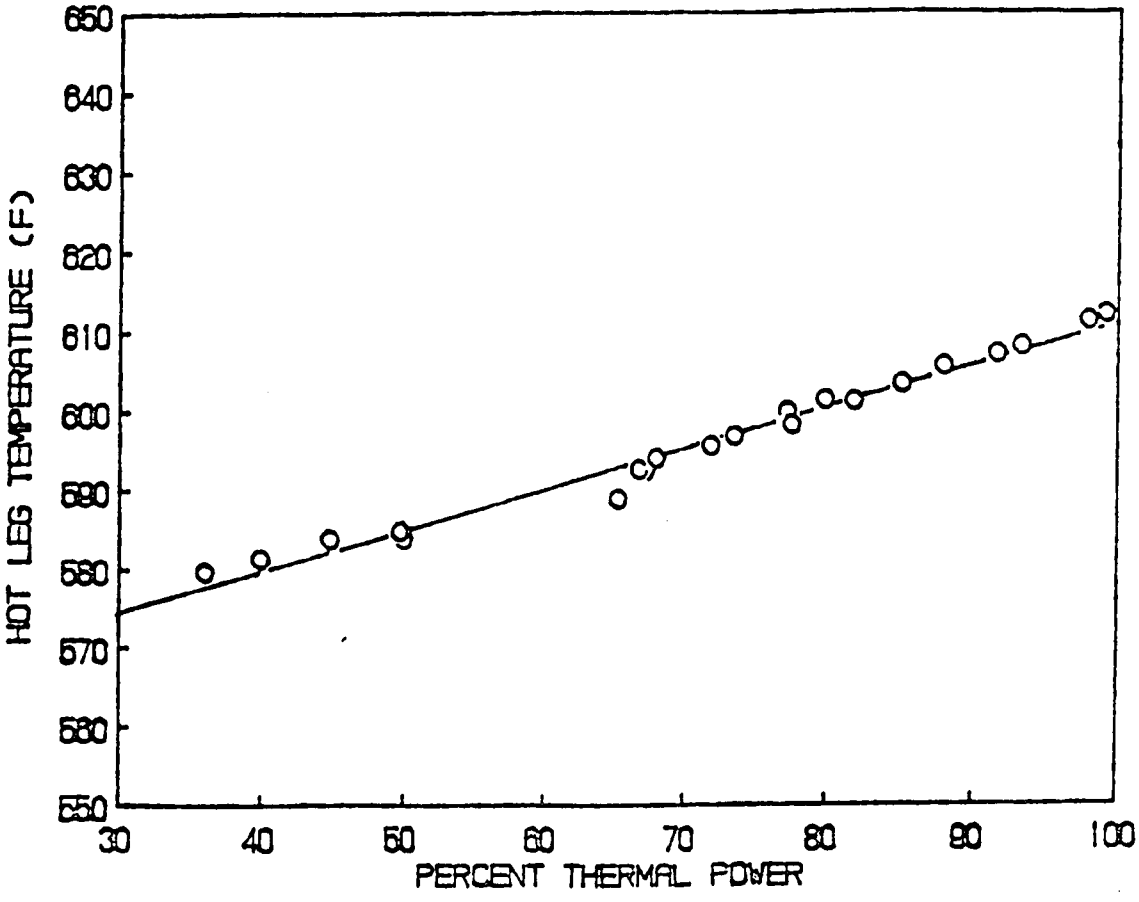

Figure 14. Hot leg temperature as a function of power for a typical steam generator.

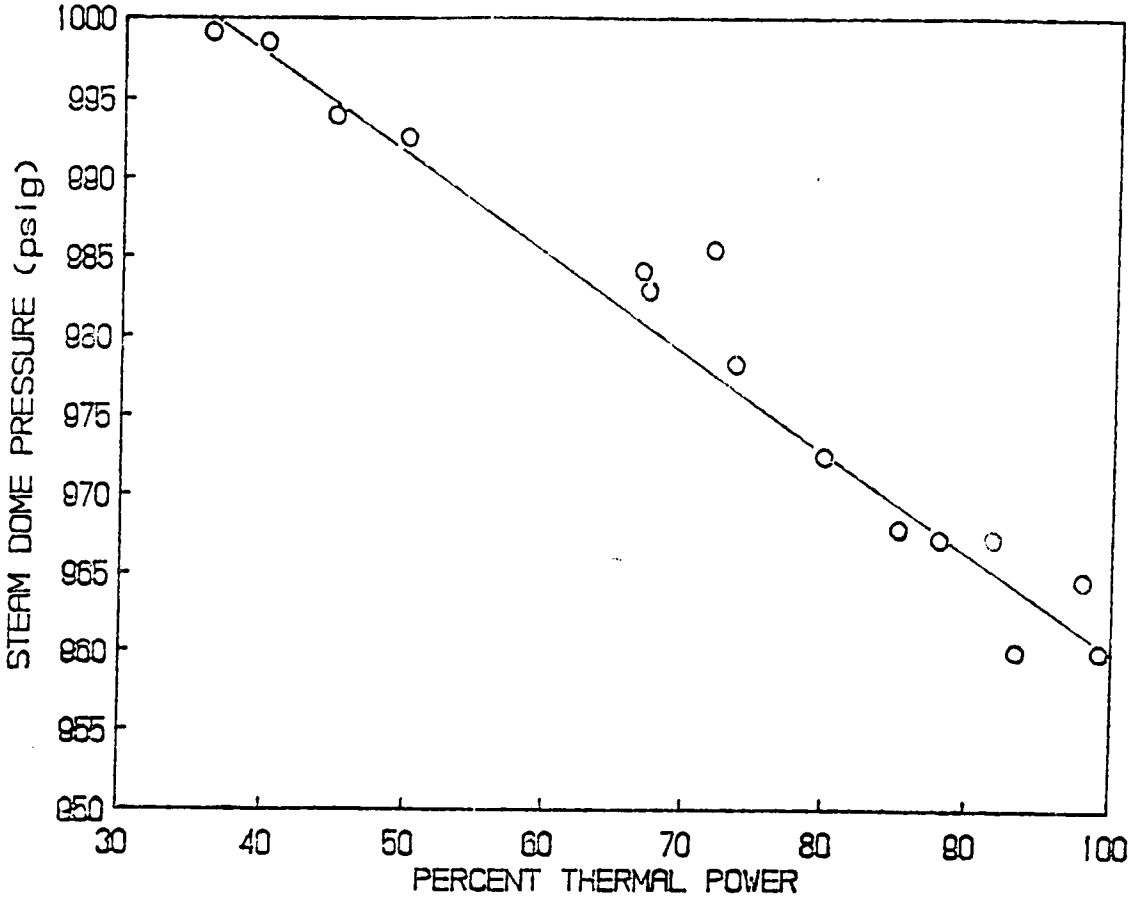

Figure 15. Steam dome pressure as a function of power for a typical steam generator.

Figure 10 is a plot of the steam generator level versus power. The steam generator level is the only controlled parameter among the six parameters. Below 66 percent power it is maintained at a value of approximately 48 feet.Above that power, the level is maintained at approximately 56 feet.

Figures 11 and 12 are plots of the primary coolant flow rate and the cold leg temperature versus power, respectively. The primary flow rate and the cold leg temperature were approximately constant over the entire power range.

Figures 13 and 14 are plots of the feedwater flow rate and hot leg temperature versus power, respectively. These two parameters were proportional to the power level of the system.

Figure 15 is a plot of the steam dome pressure versus power. The steam pressure at the dome decreased as the power increased.

The sensitivity of the measured parameters to a change in power has been calculated from this data. The sensitivity coefficients are estimations of the expected amount of change of a measured parameter for a known amount of change of an independent variable. The coefficients were calculated by dividing the difference of the sensor's average output at thirty percent power minus the value at one hundred percent power by thirty minus one hundred.

The independent variable for primary coolant flow rate and the cold leg temperature is NIL. The NIL variable is used to inform the expert system that the sensor's average output is not expected to change under normal operating conditions. Feedwater flow and steam flow rate are used as two independent parameters. The remaining parameters are dependent on the power level of the system. It is assumed that the plant will be operating under the same basic conditions prevalent when the data were acquired. Figure 16 gives the values of the sensitivity coefficients for the sensors used to monitor the steam generator.

The data were also used to set the limits of normal operation for the measured parameters. The primary flow rate and the cold leg temperature have the narrowest set of limits because they are fairly constant with respect to the reactor power. The steam generator levels have two sets of limits. This is a result of the control scheme used on that parameter. The steam generator level is usually controlled to one level below a set power level and to a higher level beyond that power level.

The other limits for the parameters are set using the value at 30 percent power and the value at 100 percent power.

4.2 The Steam Generator Knowledge Base

The steam generator chosen for the implementation is monitored with twelve sensors located in six location regions. The possible sensor location regions for the steam generator are

1. feed water inlet,
2. downcomer,
3. steam dome,
4. steam outlet,
5. cold leg, and
6. hot leg.

Figure 16 lists the variables required to represent the sensors used to monitor a typical PWR steam generator. The actual computer representations are listed in Appendix A.

NAME	mssp514	fws1517	fwsf511	rcst413b	rcst413a	cvrc1f1
LOCATION	dome	downcomer	feedwater inlet	cold-leg	hot-leg	cold-leg
TYPE	pressure	level	flow	temp.	temp.	flow
UPPER-LIMIT-1	1000.0	50.0	1415.0	558.0	615.0	111.9
LOWER-LIMIT-1	995.0	48.0	3744.0	552.5	575.0	110.6
UPPER-LIMIT-2	1000.0	59.0	1415.0	558.0	613.0	111.9
LOWER-LIMIT-2	955.0	56.5	3744.0	552.5	575.0	111.6
LIMIT 0	100	66	100	100	100	100
SENSITIVE-TO	power	nil	fwsf510	nil	power	power
SENSITIVITY	-0.618	0.0	1.0	0.0	.518	-0.1
MAXIMUM-FLUCTUATION	11.97	.156	110.01	3.45	.345	.09
MINIMUM-FLUCTUATION	1.33	1.41	12.2	.38	.383	.84
UNITS	psig	feet	lbm/sec	degF	degF	thou. gal/min

Figure 16. Values typical of computer representations for sensors used to monitor a steam generator.

CHAPTER V

RESULTS OF IMPLEMENTATION OF THE EXPERT SYSTEM

5.1 Initial Testing of the Expert System and Results

The signal validation portion of the system was tested for each of the twelve sensors used to monitor the steam generator. The tests were run with data used to simulate an increase in power from approximately 78 percent power to approximately 94 percent power after the time of the previous evaluation and before the time of the current evaluation.

Three steam dome pressure sensors were evaluated. The evaluation of two of the pressure sensors returned that the sensors were operational, but in both cases some disagreement among the redundant sensor's noise level was noted. An evaluation of the third pressure sensor revealed that this sensor had a higher noise level than the other two. The system specific limits for the expected noise level were not violated. The expert system listed the third sensor as having a possible anomaly due to improper noise with a confidence factor of 75 percent.

The evaluation of the primary coolant flow rate sensor did not have any redundant sensors for comparison. The results of the evaluation are entirely a result of the system specific tests. The expert system determined that the sensor had a possible anomaly due to an incorrect

response to a system change. The flow rate decreased approximately 0.62 thousand gallons per minute over the range of the power increase. The primary coolant flow rate was considered to be constant because its dependence on power was shown to be small and was considered negligible. The system's tolerances allowed detection of the change, and the sensor was listed as having a possible anomaly due to incorrect response to a system change.

The analysis of the feedwater flow and the steam flow sensors also do not have any redundancy to assist in the evaluation. These two sensors are expected to change the same way. Thus, for the analysis of these sensors the sensitivity analysis becomes an important test. The feedwater flow sensor's output increased approximately 813.8 lbm/sec and the steam flow sensor's output increased approximately 815.8 lbm/sec. The observed changes are well within the tolerances set by the overall accuracy of the sensor loops and the two sensors were listed as operational due to the fact that no evidence of a possible anomaly was detected.

The analysis of the hot leg temperature sensor indicated that the sensor had a possible anomaly due to an incorrect response to normal process noise. The lower limit of the expected standard deviation of the sensor's output was approximately 0.38 deg F, and the actual value was approximately 0.34 deg F. The limit was set by considering any observed standard deviation smaller than

one third or greater that three times the standard deviation calculated from a larger data file of that sensor's output. In this case the sensor is most likely operational, but the system must use the prescribed limits and in this case has no redundant sensors to assist in the evaluation. The range of normal operation for noise level was arbitrarily set for demonstration purposes. This range may be expanded or narrowed as needed.

The analysis of the cold leg temperature sensor again did not have any redundant sensors to help with the comparison, but all the system specific tests passed and the sensor was reported as operational.

The analysis of the level sensors revealed a problem associated with the expert system. There are four level sensors used to monitor the steam generator level. Three of the sensors are narrow range detectors and the other is a wide range detector. Wide range level detectors are used to measure the height of the column of fluid and narrow range detectors measure only an upper portion of the column of fluid. The narrow range level detectors tend to be more accurate and have less noise than do the wide range level detectors. The analysis of two narrow range detectors indicated that the sensors were all operational but that there was disagreement among the redundants with regard to sensor average output and noise level. The evaluation of the third narrow range detector indicated the sensor was operational but there was some disagreement among the

redundant sensors in average sensor output, noise level, and change in average sensor output. The evaluation of the wide range detector indicated that the sensor was in disagreement with three agreeing redundant sensors as far as average sensor output and sensor output noise level was concerned. The limit test also failed for the wide range level detector. The problem is that the wide range and the narrow range level detectors are redundant sensors, but they are not similar. They are two slightly different instruments used to measure two slightly different parameters, therefore their behavior is not expected to be the same. The system specific information can be corrected to include a larger range of expected outputs and noise levels, but comparison among redundants will continue to flag the sensor as possibly anomalous. One option to correct for this problem would be to consider the two types of sensors as different and thus not redundant. Another approach would be to expand the errors of all the level sensors to that of the wide range level detector and thus allow the system to look for a more disperse grouping of sensor characteristics.

Another problem illustrated by the level detectors is that of the sensitivity analysis failing for sensors that measure parameters that are sensitive to nothing, but are controlled at different values at different power levels. If a power change crosses the power level at which the parameter is controlled the sensor output will change. The

sensitivity analysis will expect the average sensor output not to change and will fail upon the observed change. The steam generator level is controlled to change only when the power gets above a certain value or drops below it again in a Westinghouse type PWR. When this point is passed between evaluations the following evaluation will show the sensor's output to change considerably, but will expect it not to change at all and the sensitivity test will fail. The system must be altered to account for this possible scenario.

5.2 Secondary Testing of the Expert System and Results

In order to correct the problems arising from different characteristics of the wide and narrow range level detectors the wide range detector has been given a new type name to distinguish it as a different type of sensor loop. The limits of normal operation have been expanded for the wide range level detector to account for observed behavior in the sensor loop.

The tests were run again for the four level detectors. Two of the narrow range level detectors evaluated were found to be normal; however, there was disagreement with one redundant sensor's amount of change in average output. Upon analysis of the third sensor it was observed that the average sensor output of level sensor fws1519 increased approximately 0.35 feet when the average output of the redundant sensors showed no significant change. As a

result of this, the expert system listed sensor fwsl519 as having a possible anomaly due to an incorrect response to a system change.

The evaluation of the wide range level detector concluded that the sensor was operational because all of the expected system specific characteristics were satisfied.

The system specific test to perform sensitivity analysis was revised to properly handle the situation of a controlled parameter changing upon a system change when the parameter is not sensitive normally to any independent parameter. The routine will return "true" if the power change since the previous evaluation crossed the sensor's limit0 variable and the sensor is not sensitive to anything.

The evaluation that is performed by the expert system was designed to mimic an operator's method of signal validation by using all available information to access as sensor's performance. The system locates redundant sensors, and uses comparison against them to evaluate a sensor and uses specific knowledge of the sensor and the system that the sensor is monitoring to predict expected sensor behavior and compares the expected behavior to the observed behavior. The evaluation uses both sources of information to reach a conclusion about a sensor, and has the ability to overrule a test if there is strong evidence from the redundant sensors that the sensor is operational

even though that test indicated that the sensor is possibly anomalous.

The system specific test used to predict the expected change in sensor output proved to be very accurate. The predicted changes in average sensor output for a change in the operating condition of the plant were within overall error tolerances for all but one sensor. A comparison among the redundant sensors showed that the sensor was in disagreement with two consistent sensors.

The limits of the expected noise level in a sensor's output usually have a large range. For example a feedwater flow sensor could have an output with a standard deviation about the mean of 12.1 to 110 lbm/sec and could be considered normal. While the test is not extremely precise, it does accomplish the task for which it is designed. It allows the expert system to determine if a sensor's output is responding to normal process noise or if a sensor's output is passive.

Appendix B contains a listing of a sample consultation of the expert system. The system first provides information about a sensor, then performs an evaluation of the sensor's performance and lists the results.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE WORK

6.1 Conclusions

The signal validation and sensor information expert system has been implemented and tested. The expert system interactively provides the user with a list of sensors used to monitor a subsystem of a power plant and allows the user to ask questions about the characteristics of those sensors.

The expert system also performs an evaluation of a sensor's current output characteristics. The evaluation is fully automated and is designed to mimic an operator's method of signal validation. The evaluation was tested using operational data from twelve of the sensors used to monitor a steam generator in an operating commercial nuclear power plant. The results of the testing demonstrated successfully that the evaluation detected three possibly anomalous sensor behaviors, and was able to handle conflicting information from the comparison among redundant sensors and the system specific test used to detect each of the three possible anomalies.

This system performs its evaluation effectively with a minimum of calculation, just as an operator would. The system uses comparisons among redundant sensors, when they are available, and combines the knowledge about the

expected behavior of each sensor to perform a complete evaluation of a sensor's performance.

6.2 Recommendations for Future Work

The current system requires examination of a sensor's output over its entire range of operation in order to determine the system specific information needed to predict the expected sensor behavior. The system is designed to validate sensors only during steady-state operation and under normal control strategy. A natural extension of this validation concept would be an analysis that predicts relationships among sensor outputs based on a model of the plant and a knowledge of the basic principles of thermodynamics.

Using its knowledge of the subsystem, the sensors used to monitor the subsystem, a few basic principles of thermodynamics, and the observed output of other sensors the expert system would be able to predict a sensor's expected output. This prediction would be independent of any control action used on the system.

The expert system would have the ability to trace and sum flow rates, and temperature and pressure changes throughout the plant. The system would be able to predict sensor output in relation to sensors located upstream and downstream.

An extension of this approach to the current signal validation expert system would allow the system to further mimic an operator's ability to perform signal validation; however, the validation would no longer be restricted to normal control strategies.

REFERENCES

REFERENCES

1. Finch, F. E., Kramer, M. A., "Expert System Methodology for Process Malfunction Diagnosis", Presented at tenth IFAC World Congress on Automatic Control, July 1987.

2. Washio, T., et al, "Automated Derivation of Failure Symptoms for Diagnosis of a Nuclear Power Plant", Ann. Nucl. Energy. Vol. 13, No. 8, pp 459-465, 1986.

3. Washio, T. et al, "Semantic Approach to Automated Failure Diagnosis in Nuclear Power Plants", Presented at Sixth Power Plant Dynamics, Control and Testing Symposium, Vol. 1, pp 39-42, April 1986.

4. Kramer, M. A., Palowitch, B. L., "Expert System and Knowledge-Based Approaches to Process Malfunction Diagnosis," Presented at AIChE National Meeting, Chicago, Nov. 1985.

5. Bastle, W., Felkel, L. Proc. ANS/ENS Topic. Mtg on Thermal Reactor Safety, Knoxville, TN, p 285, 1980.

6. Meijer, C., Frogner, B,On-line Power Plant Alarm and Disturbance Detection System," Report EPRI NP-1379, 1980.

7. Upadhyaya, B. R. et al, "Development of an Integrated Signal Validation Systems in Nuclear Power Plants," Annual Report prepared for the U.S. Department of Energy by the University of Tennessee, DOE/NE/37959-12, Sept. 1987.

8. Winston, P.H., Horn, B.K.P., Lisp, Addison- Wesley, 1984.

9. O'Shea, T., Eisenstadt, M. "Artificial Intelligence Tools, Techniques, and Applications," Harper and Row, New York, 1984.

APPENDICES

APPENDIX A

PROGRAM LISTING

```
;*****************************************************
*******************
;*****************************************************
*******************
;                    SENSOR INFORMATION AND SIGNAL VALIDA
TION
;                                 EXPERT SYSTEM
;*****************************************************
*******************
; DEVELOPED BY:  L. QUALLS
;*****************************************************
*******************



;*****************************************************
*******************
;DEFINE NECESSARY STRUCTURE AND CALL-OUT ROUTINES TO EX
TERNAL PROGRAMS
;*****************************************************
*******************
;DEFINE SENSORLOOP STRUCTURE
(defstruct sensorloop
  name kind location filename result-filename
  current-reading previous-reading change units maximum
-fluctuation
  minimum-fluctuation fluctuation sensitive-to error
  sensitivity limit1 upper-limit1 lower-limit1 upper-li
mit2 lower-limit2)


;*****************************************************
*******************
;DEFINE CALL-OUT ROUTINE AVERAGE
(define-external-routine (average
                              :file "average"
                              :result (:lisp-type single
-float
                                                  :vax-t
ype :f-floating)
                              )
  (filename :lisp-type string
            :vax-type :asciz
            :access :in-out))


;*****************************************************
*******************
;DEFINE CALL-OUT ROUTINE FLUCTUATION
(define-external-routine (fluctuation
                              :file "fluctuation"
                              :result (:lisp-type single
-float
                                                  :vax-t
```

```
)
     (format T"~%The redundant sensors are ~d" redundan
t))
    ((eq action 4)(select-sensor)(find-relations suspec
t)
     (format T"~%The average output of the redundant se
nsors is ~d"
             (ave redundant)))
    ((eq action 5)(select-sensor)(find-location sensorl
oops))
    ((eq action 6)(find-type sensorloops))
    ((eq action 7)(find-location* sensorloops))
    ((eq action 'q) (return t))
    (t (format T"~%that is not a valid option please tr
y again~%")))
  (go loop)))

;*****************************************************
*******************
;UPDATE THE CHARACTERISTICS OF A SENSOR THAT CAN CHANGE
(defun update(sensor1)
  (setq sensor2 (eval sensor1))
  (setf (sensorloop-previous-reading sensor2 )
        (sensorloop-current-reading sensor2))
  (setf (sensorloop-current-reading sensor2 )
        (float(call-out average (sensorloop-filename se
nsor2))))
  (setf (sensorloop-change sensor2)(- (sensorloop-curre
nt-reading sensor2)
                                      (sensorloop-previ
ous-reading sensor2)))
  (setf (sensorloop-fluctuation sensor2 )
        (sqrt(float(call-out fluctuation (sensorloop-fi
lename sensor2)))))
  )

;*****************************************************
*******************
;SELECT A SENSOR FOR EVALUATION
;*****************************************************
*******************
(defun select-sensor()
  (format T"~%Enter the sensor you wish to have examine
d~%")
  (cond((not(null suspect))
        (format T"~%If you wish to continue examining t
he same sensor type  y~%
")))
  (setq susp (read))
  (cond((eq susp 'y) nil)
       ((null (member susp sensorloops))
        (format T"~%That is not a valid sensor name")
        (select-sensor))
       (t(setq suspect (eval susp)))))
```

```
;*****************************************************
*******************
;SHOW THE USER THE CHARACTERISTICS OF A SELECTED SENSOR
(defun tell()
  (prog(action)
    loop
    (format T"~%the following are available characteris
tics of the sensorloop")
    (format T"~% 1  current-reading")
    (format T"~% 2  previous-reading")
    (format T"~% 3  change from previous evaluation")
    (format T"~% 4  upper and lower limits of normal op
eration for current power")
    (format T"~% 5  the limits of expected fluctuation
of sensor output about the mean")
    (format T"~% 6  the fluctuation observed during las
t data acquistion")
    (format t"~% 7  which parameter this sensor is expe
cted to change with")
    (format T"~% 8  the expected change per unit for a
change in the indepentant variable")
    (format T"~% 9  sensor location")
    (format T"~% 10  sensor type")
    (format T"~% 11 units")
    (format T"~% q quit~%")
    (format T"~%Please enter the appropriate number for
 the characteristic you need  ")
    (setq action (read))
;RETURN THE REQUESTED INFORMATION
    (cond((eq action 1)(format T" ~d ~d"(sensorloop-cur
rent-reading suspect)
                               (sensorloop-units suspec
t)))
         ((eq action 2)(format T" ~d ~d"(sensorloop-pre
vious-reading suspect)
                               (sensorloop-units suspec
t)))
         ((eq action 3)(format T" ~d ~d"(sensorloop-cha
nge suspect)
                               (sensorloop-units suspec
t)))
         ((eq action 4)(cond((< current-power (sensorlo
op-limit1 suspect))
                             (format T" ~d ~d"
                                     (sensorloop-lower-
limit1 suspect)
                                     (sensorloop-upper-
limit1 suspect)))
                            (t(format T" ~d ~d"
                                      (sensorloop-lower
-limit2 suspect)
                                      (sensorloop-upper
```

```
limit2 suspect))))
          (format T" ~d" (sensorloop-units suspect)))
         ((eq action 5)(format T" ~d ~d and ~d ~d "
                               (sensorloop-maximum-fluc
tuation suspect)
                               (sensorloop-units suspec
t)
                               (sensorloop-minimum-fluc
tuation suspect)
                               (sensorloop-units suspec
t)))
         ((eq action 6)(format T" ~d ~d"(sensorloop-flu
ctuation suspect)
                               (sensorloop-units suspec
t)))
         ((eq action 7)(format T" ~d "(sensorloop-sensi
tive-to suspect)))
         ((eq action 8)(format T" ~d "(sensorloop-sensi
tivity suspect))
          (format T" unit change per unit change of ind
ependant parameter"))
         ((eq action 9)(format T" ~d "(sensorloop-locat
ion suspect)))
         ((eq action 10)(format T" ~d "(sensorloop-kind
 suspect)))
         ((eq action 11)(format T" ~d "(sensorloop-unit
s suspect)))
         ((eq action 'q)(return t))
         (t(format T"~% that is not a valid option")(te
ll)))
    (go loop)))

;*****************************************************
*******************
;DETERMINE HOW A SENSOR IS RELATED TO OTHERS
(defun find-relations (s)
  (fr s sensorloops))

(defun fr (sensor3 sensors)
  (cond
    ((null sensors) T)
    (T (cond
         ((null sensors) nil)
         ((cond
            ((string= (sensorloop-name sensor3)
                      (sensorloop-name (eval (car senso
rs)))) nil)
            ((eq(- (eval(sensorloop-location sensor3))
                   (eval(sensorloop-location (eval(car
sensors))))) 0)
             (cond((string= (sensorloop-kind sensor3)
                            (sensorloop-kind (eval(car
sensors))))
                  (setq redundant (push1 redundant (li
```

```
t(car sensors)))))
                  (T(setq same-location (push1 same-loc
ation (list(car sensors))
                                                  )))))
            ((eq (- (eval(sensorloop-location sensor3))
                    (eval(sensorloop-location (eval(car
 sensors))))) -1)
             (setq upstream (push1 upstream (list(car s
ensors))))
             (cond ((string= (sensorloop-kind sensor3)
                             (sensorloop-kind (eval(car
 sensors))))
                    (setq upstream-red (append upstream
-red (list(car sensors))
                                                   )))))
            ((eq (- (eval(sensorloop-location sensor3))
                    (eval(sensorloop-location (eval(car
 sensors))))) 1)
             (setq downstream (push1 downstream (list(c
ar sensors))))
             (cond ((string= (sensorloop-kind sensor3)
                             (sensorloop-kind (eval(car
 sensors))))
                    (setq downstream-red (append downst
ream-red (list(car sensors)

           ))))))
            (T nil))))
       (fr sensor3 (cdr sensors)))))



;*****************************************************
*******************
;FIND ALL SENSORS AT A SPECIFIED LOCATION
(defun find-location*(sensors)
  (format T"~%the possible sensor locations are~%")
  (print locations)
  (format T"~%From what location do you wish to have a
list of sensors~%")
  (format T"~% enter location ~%")
  (setq loca (read))
  (prog(same-location)
    loop
    (cond((null sensors)(format T"~%These sensors are l
ocated
at the ~d ~d~%"loca  same-location)
          (return T))
         ((string= loca
                   (sensorloop-location (eval(car senso
rs))))
          (setq same-location (push1 same-location (lis
t(car sensors))))
          (setq sensors (cdr sensors))(go loop))
```

```
          (T (setq sensors (cdr sensors))(go loop)))
     (go loop))
   )


;*****************************************************
*******************
;FIND ALL SENSORS LOCATED IN THE SAME REGION AS ANOTHER
(defun find-location(sensors)
  (prog(same-location)
    loop
    (cond((null sensors)(format T"~%These sensors are l
ocated
at the same location as ~d ~d~%"(sensorloop-name suspec
t)  same-location)
          (return T))
         ((string= (sensorloop-location suspect)
                   (sensorloop-location (eval(car senso
rs))))
          (setq same-location (push1 same-location (lis
t(car sensors))))
          (setq sensors (cdr sensors))(go loop))
         (T (setq sensors (cdr sensors))(go loop)))
    (go loop))
  )


;*****************************************************
*******************
;FIND SENSORS OF A SPECIFIED TYPE
(defun find-type(sensors)
  (format T"~% These are the type of sensors used to mo
nitor the system~%")
  (print sensor-types)
  (format T"~% what type of sensors do you wish to have
 a list of ")
  (format T"~% enter type of sensor~%")
  (setq typeof (read))
  (prog(same-types)
    loop
    (cond((null sensors)(format T"~%These sensors are a
ll of the ~d sensors ~d~%"
                                typeof same-types)(retu
rn T))
         ((string= typeof
                   (sensorloop-kind (eval(car sensors))
))
          (setq same-types (push1 same-types (list(car
sensors))))
          (setq sensors (cdr sensors))(go loop))
         (T (setq sensors (cdr sensors))(go loop)))
    (go loop))
  )
```

```
;*****************************************************
*********************
;SIGNAL VALIDATION PORTION OF THE PROGRAM
;*****************************************************
*********************
;THE MAIN CALLING ROUTINE OF THE SIGNAL VALIDATION PORT
ION OF THE PROGRAM
 (defun advise()
  (setq red-groups1 (redundant-test redundant 'sensorlo
op-current-reading))
  (regroup)
  (setq red-groups2 (redundant-test redundant 'sensorlo
op-change))
  (regroup)
  (setq red-groups3 (redundant-test redundant 'sensorlo
op-FLUCTUATION))
  (regroup)
  (setq red-agree1 (agree suspect redundant 'sensorloop
-current-reading))
  (setq red-agree2 (agree suspect redundant 'sensorloop
-change))
  (setq red-agree3 (agree suspect redundant 'sensorloop
-fluctuation))
  (resetcf)
  (diagnosecf)
  (learn)
  (results)
  )


;*****************************************************
*********************
;CALCULATE VARIABLES NEEDED TO PERFORM COMPARISON AMONG
 REDUNDANTS
;*****************************************************
*********************

;*****************************************************
*********************
;CALCULATE THE RATIO OF AGREEING SENSORS TO SENSORS IN
LIST
;*****************************************************
*********************
(defun agree (sensor4 sensors1 parameter)
  (setq total (length sensors1))
  (cond((eq total 0)
        (setq value -1))
       (T
           (agree2 sensor4 sensors1 parameter)
           (setq value (/ number total))))
  (setq number 0.0)
  (/ value 1))
```

```
(defun agree2(sensor4 sensors1 parameter)
  (cond((eq (length sensors1) 0.0) nil)
       (T
          (setq sensors (eval `(car sensors1)))
          (cond((eqtol (eval `(,parameter sensors))
                       (eval `(,parameter suspect)))
                (setq number (+ number 1))))))
  (cond((null(cdr sensors1)) nil)
       (T
          (agree2 sensor4 (cdr sensors1) parameter))))

;*****************************************************
*******************
;DETERMINE THE NUMBER OF GROUPS IN AGREEING IN THE REDU
NDANT LIST
;*****************************************************
*******************
(defun redundant-test(alist parameter)
  (cond((null alist) nil)
       ((null group1)(setq group1 (list(car alist)))
        (redundant-test (cdr alist) parameter))
       ((eqtol (eval `(,parameter (eval(car group1))))
               (eval `(,parameter ,(car alist))))
        (setq group1
              (append group1 (list(car alist))))(redund
ant-test (cdr alist)

         parameter))
       ((null group2)(setq group2 (list(car alist)))
        (redundant-test (cdr alist) parameter))
       ((eqtol (eval `(,parameter (eval(car group2))))
               (eval `(,parameter ,(car alist))))
        (setq group2
              (append group2 (list(car alist))))(redund
ant-test (cdr alist)

         parameter))
       ((null group3)(setq group3 (list(car alist)))
        (redundant-test (cdr alist) parameter))
       ((eqtol (eval `(,parameter (eval(car group3))))
               (eval `(,parameter ,(car alist))))
        (setq group3
              (append group3 (list(car alist))))(redund
ant-test (cdr alist)

         parameter))
       ((null group4)(setq group4 (list(car alist)))
        (redundant-test (cdr alist) parameter))
       ((eqtol (eval `(,parameter (eval(car group4))))
               (eval `(,parameter ,(car alist))))
        (setq group4
              (append group4 (list(car alist))))(redund
ant-test (cdr alist)
```

```
          parameter))
        ((null group5)(setq group5 (list(car alist)))
         (redundant-test (cdr alist) parameter))
        ((eqtol (eval `(,parameter (eval(car group5))))
                (eval `(,parameter ,(car alist))))
         (setq group5
               (append group5 (list(car alist))))(redund
ant-test (cdr alist)

          parameter))
        (T (print"fell thru")))
  (cond((null group1)(setq result 0))
       ((null group2)(setq result 1))
       ((null group3)(setq result 2))
       ((null group4)(setq result 3))
       ((null group5)(setq result4))
       (T (setq result 5))))


;*****************************************************
*******************
;BEGIN INFERENCE ENGINE
;*****************************************************
*******************
;*****************************************************
*******************
(DEFUN DIAGNOSECF()
  (PROG (POSSIBILITIES ASKED)
    (SETQ POSSIBILITIES hypothesis)
    LOOP
    (COND((NULL POSSIBILITIES)(PRINT POSSIBILITIES)
          (ORDER POSS)(return nil))
         ((VERIFYCF (car(CAR POSSIBILITIES)))
          (PRINT (CAR POSSIBILITIES))
          (RETURN T)))
    (SETQ POSSIBILITIES (CDR POSSIBILITIES))
    (GO LOOP)))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN REMEMBER(NEW)
  (COND ((MEMLIST NEW FACTS)NIL)
        (T(SETQ FACTS (CONS NEW FACTS))
          NEW)))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN RECALL (FACT)
  (COND((MEMLIST FACT FACTS) FACT)
```

```
        (T NIL)))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN TESTIF (RULE)
  (PROG (IFS)
    (SETQ IFS (CDADDR RULE))
    LOOP
    (COND ((NULL IFS)(RETURN T))
          ((RECALL (CAR IFS)))
          (T (RETURN NIL)))
    (SETQ IFS (CDR IFS))
    (GO LOOP)))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN USETHENCF(RULE)
    (format t"~%IT HAS BEEN DEDUCED BY USE OF RULE ~d t
hat ~%~d~%has confidence
~d" (CADR RULE) (caadr (cadddr rule)) (cadadr (cadddr r
ule)))
    (march (CDR (CADDDR RULE)))
    )
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN INTHENCF(FACT)
  (MAPCAN #'(LAMBDA (R)
              (COND ((THENPCF FACT R)
                     (SETQ MAPVAR (APPEND MAPVAR (LIST
R))))))
  RULES))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN THENPCF (FACT RULE)
  (MEMLIST  fact (cadr(CADDDR RULE))))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN TRYRULECF(RULE)
  (AND(TESTIF RULE)(USETHENCF RULE)))

;*****************************************************
```

```
********************
(DEFUN TRYRULE+CF (RULE)
  (AND (TESTIF+CF RULE) (USETHENCF RULE)))

;*****************************************************
********************
(DEFUN STEPFORWARDCF()
  (PROG (RULELIST)
    (SETQ RULELIST RULES)
    LOOP
    (COND ((NULL RULELIST)(RETURN NIL))
          ((TRYRULECF (CAR RULES))(RETURN T)))
    (SETQ RULELIST (CDR RULELIST))
    (GO LOOP)))

;*****************************************************
********************
(DEFUN DEDUCECF()
  (PROG (PROGRESS)
    LOOP
    (COND ((STEPFORWARDCF)(SETQ PROGRESS T))
          (T (RETURN PROGRESS)))
    (GO LOOP)))

;*****************************************************
********************
(DEFUN TESTIF+CF (RULE)
  (PROG (IFS)
    (SETQ IFS (CDADDR RULE))
    LOOP
    (COND ((NULL IFS)(RETURN T))
          ((VERIFYCF (CAR IFS)))
          (T(RETURN NIL)))
    (SETQ IFS (CDR IFS))
    (GO LOOP)))

;*****************************************************
********************
(DEFUN VERIFYCF(FACT)
  (PROG (RELEVANT1 RELEVANT2)
    (COND ((RECALL FACT)(RETURN T)))
    (setq mapvar nil)
    (INTHENCF FACT)
    (SETQ RELEVANT1 MAPVAR)
    (SETQ RELEVANT2 RELEVANT1)
    (COND((NULL RELEVANT1)
          (COND((MEMLIST FACT ASKED)(RETURN NIL))
               ((exam FACT)
                (REMEMBER FACT)
                (SETQ ASKED (CONS FACT ASKED))
                (RETURN T))
               )))
    LOOP1
    (COND ((NULL RELEVANT1)(GO LOOP2))
```

```
        ((TRYRULECF (CAR RELEVANT1))(RETURN T)))
    (SETQ RELEVANT1 (CDR RELEVANT1))
    (GO LOOP1)
    LOOP2
    (COND ((NULL RELEVANT2)(GO EXIT))
        ((TRYRULE+CF (CAR RELEVANT2))(RETURN T)))
    (SETQ RELEVANT2 (CDR RELEVANT2))
    (GO LOOP2)
    EXIT
    (RETURN NIL)))
;*****************************************************
*******************

;*****************************************************
*******************
;SHOW THE USER ALL CONCLUSIONS REACHED BY THE INFERENCE
 ENGINE
;*****************************************************
*******************

;*****************************************************
*******************
(defun showem (alist)
  (cond((null alist) nil)
       ((equal (cadr alist) 0.0) nil)
       (t(format T"~% conclusion ~d has confidence ~d~%
" (car alist)(cadr alist))))
  (cond((null alist) nil)
       (T (showem (cddr alist)))))
;*****************************************************
*******************

;*****************************************************
*******************
;RECALCULATE THE CONFIDENCE FACTORS
;*****************************************************
*******************
(DEFUN CHANGE(NEW)
  (SETQ CFP 0.0)
  (SETQ CFP (EVAL(CAADR (ASSOC (CAR NEW) POSS :TEST 'EQ
UAL))))
  (SETQ CFN 0.0)
  (SETQ CFN (CAADR NEW))
  (SET (CAADR(ASSOC (CAR NEW) POSS :TEST 'EQUAL))
       (+ CFP (* CFN (- 1.0 CFP)))))
;*****************************************************
*******************
;PUT THE FINAL CONCLUSIONS IN THE ORDER OF HIGHEST CONF
IDENCE
;*****************************************************
*******************
(DEFUN MARCH(GP)
  (SETQ COP GP)
  (COND
```

```
    ((NULL COP)NIL)
    (T(CHANGE (CAR COP))
      (SETQ COP (CDR COP))
      (MARCH COP))))
;*****************************************************
*******************

;*****************************************************
*******************
(DEFUN ORDER(WORK)
  (cond((null work)(format T"~% CONCLUSIONS IN DECENDIN
G PRIORITY"))
       ((null templ)(setq templ (append (list(caar work
))
                                        (list(eval(caadar
 work)))))
        (setq work (cdr work))(order work))
       (t(cond((< (eval(caadar work)) (cadr templ))
               (setq templ (append templ (append (list(
caar work))
                                                 (list(ev
al

 (caadar work)))))
               (setq work (cdr work))
               (order work))
              (t(setq templ (append (append (list(caar
work))
                                           (list(eval(c
aadar work))))
                                    templ))
                (setq work (cdr work))(order work))))))



;*****************************************************
*******************
;SYSTEM SPECIFIC TESTS USED TO PERFORM THE EVALUATION
;*****************************************************
*******************

;*****************************************************
*******************
(defun sensitivity-check(sensor)
  (cond((or(and(null (sensorloop-sensitive-to suspect))
               (< current-power (sensorloop-limit1 susp
ect))
               (> previous-power (sensorloop-limit1 sus
pect)))
           (and(null (sensorloop-sensitive-to suspect))
               (> current-power (sensorloop-limit1 susp
ect))
```

```
                (< previous-power (sensorloop-limit1 sus
pect)))) T)
       ((null (sensorloop-sensitive-to sensor))
        (eqtol 0.0 (sensorloop-change sensor)))
       ((string= (sensorloop-sensitive-to sensor) 'powe
r)
        (eqtol
            (* (- current-power previous-power) (sensor
loop-sensitivity sensor))
            (sensorloop-change sensor)))
       (T (eqtol (* (- (sensorloop-previous-reading
                           (eval (sensorloop-sensitive-
to sensor)))
                       (sensorloop-current-reading
                           (eval(sensorloop-sensitive-t
o sensor))))
                   (sensorloop-sensitivity sensor))
                (sensorloop-change sensor)))))
;*****************************************************
*******************

;*****************************************************
*******************
(defun expected-change(sensor)
  (cond((null (sensorloop-sensitive-to sensor)) 0.0)
       ((string= (sensorloop-sensitive-to sensor) 'powe
r)
        (* (- current-power previous-power) (sensorloop
-sensitivity sensor)))
       (T (* (- (sensorloop-current-reading
                           (eval (sensorloop-sensitive-
to sensor)))
                       (sensorloop-previous-reading
                           (eval (sensorloop-sensitive-
to sensor))))
                   (sensorloop-sensitivity sensor)))))
;*****************************************************
*******************

;*****************************************************
*******************
(defun limit-check(sensor)
  (cond((< current-power (sensorloop-limit1 sensor))
       (and (>= (sensorloop-current-reading sensor)
                (sensorloop-lower-limit1 sensor))
            (<= (sensorloop-current-reading sensor)
                (sensorloop-upper-limit1 sensor))))
      ((>= current-power (sensorloop-limit1 sensor))
       (and (>= (sensorloop-current-reading sensor)
                (sensorloop-lower-limit2 sensor))
            (<= (sensorloop-current-reading sensor)
                (sensorloop-upper-limit2 sensor))))))
;*****************************************************
*******************
```

```
;*****************************************************
*******************
;ROUTINE USED TO ALLOW LEARNING OF SYSTEM SPECIFIC BEHA
VIOR
;*****************************************************
*******************
(defun learn()
  (cond((and (equal cf4 .99)(equal cf27 .99)
             (> current-power (sensorloop-limit1 suspec
t))
             (> (sensorloop-current-reading suspect)(se
nsorloop-upper-limit2

 suspect)))
       (setf (sensorloop-upper-limit2 suspect) (sensor
loop-current-reading
                                                  sus
pect)))
       ((and (equal cf4 cf27)(equal cf4 .99)
             (> current-power (sensorloop-limit1 suspec
t))
             (< (sensorloop-current-reading suspect)(se
nsorloop-lower-limit2

 suspect)))
       (setf (sensorloop-lower-limit2 suspect) (sensor
loop-current-reading
                                                  sus
pect)))
       ((and (equal cf4 cf27)(equal cf4 .99)
             (< current-power (sensorloop-limit1 suspec
t))
             (> (sensorloop-current-reading suspect)(se
nsorloop-upper-limit1)))
       (setf (sensorloop-upper-limit1 suspect) (sensor
loop-current-reading
                                                  sus
pect)))
       ((and (equal cf4 cf27)(equal cf4 .99)
             (< current-power (sensorloop-limit1 suspec
t))
             (< (sensorloop-current-reading suspect)(se
nsorloop-lower-limit1)))
       (setf (sensorloop-lower-limit1 suspect) (sensor
loop-current-reading
                                                  sus
pect))))
  (cond
    ((and (equal cf8 .99)(equal cf27 .99)(not(equal (se
nsorloop-change

 suspect) 0.0)))
     (cond((null (sensorlcop-sensitivity suspect)) nil)
```

```
          ((string= (sensorloop-sensitive-to suspect) '
power)
           (setf (sensorloop-sensitivity suspect)
                 (/ (- previous-power current-power)
                    (- (sensorloop-previous-reading sus
pect)
                       (sensorloop-current-reading susp
ect)))))
          (t(setf (sensorloop-sensitivity suspect)
                  (/ (- (sensorloop-previous-reading (s
ensorloop-sensitive-to

  suspect))
                        (sensorloop-current-reading (se
nsorloop-sensitive-to

 suspect)))
                     (- (sensorloop-previous-reading su
spect)
                        (sensorloop-current-reading sus
pect)))))))))
    (cond
      ((and (equal cf15 .99) (equal cf28 .99))
       (cond((< (sensorloop-fluctuation suspect) (senso
rloop-minimum-fluctuation
                                                  su
spect))
             (setf (sensorloop-minimum-fluctation suspe
ct)
                   (sensorloop-fluctuation suspect)))
            (t
                (setf (sensorloop-maximum-fluctuation s
uspect)
                      (sensorloop-fluctuation suspect))
)))))
;*****************************************************
*******************

;*****************************************************
*******************
;SHOW THE RESULTS
;*****************************************************
*******************
(defun results()
  (format T"~%*****************************************
**********************")
  (cond((> cf32 0.0)
        (format T"~%The sensor is operational~%~%")))
  (cond((> cf29 0.0)
        (format T"~%The sensor has failed due to incorr
ect average output.    ~d
~%~%" cf29)))
  (cond((> cf30 0.0)
        (format T"~%The sensor has failed due to an inc
```

```
rrect response to a
system change.    ~d~%~%" cf30)))
  (cond((> cf31 0.0)
       (format T"~%The sensor has failed due to an inco
rrect response to
                  normal process noise.    ~d~%~%" cf31))
)
  (format T"~%****************************************
**********************")
  (format T"~%Do you wish to see the facts and results
which lead to the
            conclusions of the evaluation ? (y/n) ~% ")
  (setq response (read))
  (cond((eq response 'y)(result*)))
  (format T"~%Please enter   y   when ready to resume t
he consultation~%~%")
  (read))



;****************************************************
********************
(defun result*()
  (cond((NULL redundant)
        (format T"~%~%There are no redundant sensors av
aiable to assist in
 the evaluation.  The final conclusions are a result of
 the system specific
 test designed to detect the failure mode~%~%")
        (ss-ave)
        (ss-change)
        (ss-noise))
       (T
           (red-ave)
           (ss-ave)
           (red-change)
           (ss-change)
           (red-noise)
           (ss-noise)
           (overide))
       ))
;****************************************************
********************
(defun red-ave()
  (format T"the redundant analysis of the average outpu
t is ")
  (cond
    ((> cf1 0.0)
     (format t" %strongly conclusive that the sensor ha
s failed"))
    ((> cf2 0.0)
     (format t" %fairly conclusive that the sensor has
failed"))
    ((> cf3 0.0)
```

```
     (format t"~%slightly conclusive that the sensor ha
s failed"))
    ((> cf5 0.0)
     (form t" %fairly conclusive that the sensor is n
ormal"))
    ((> cf6 0.0)
     (for at t" %slightly conclusive that the sensor is
 normal"))
    ((> cf4 0.0)
     (fc mat t" %strongly conclusive that the sensor is
 normal'))
    ((> cf7 0.0)
     (format T"inconclusive")))
  (for at T" because")
  (for at T"~%there are ~d redundant sensors" (length r
edunda t))
  (fo mat T"~%and there are ~d groups with the same ave
rage
  ou put within the redundant list"
          (* 1 red-groups1))
  (format T"~%and that the suspect sensor agrees with ~
d s nsors
 in the redundant-list" (* red-agree1 (length redundant
))))

;*********************************************************
*********************
 defun ss-ave()
  (format T"~%The limit checking test indicated the sen
sor was ")
  (cond((> cf27 0.0)(format T"failed"))
       (t(format T"normal")))
  (format T"~%The expected limits of operation
are ~d and ~d and the sensor had an output of ~d ~d"
          (exu1 suspect)
          (ex11 suspect)
          (sensorloop-current-reading suspect)
          (sensorloop-units suspect)))

;*********************************************************
*********************
(defun exu1(sensor)
  (cond((< current-power (sensorloop-limit1 sensor))
        (sensorloop-upper-limit1 sensor))
       (t
           (sensorloop-upper-limit2 sensor))))


;*********************************************************
*********************
(defun ex11(sensor)
```

```
  (cond((< current-power (sensorloop-limit1 sensor))
        (sensorloop-lower-limit1 sensor))
       (t
           (sensorloop-lower-limit2 sensor))))


;*****************************************************
*******************
(defun red-change()
  (format T"~%the redundant analysis of the output chan
ge is ")
  (cond
    ((> cf8 0.0)
     (format t"~%strongly conclusive that the sensor ha
s failed"))
    ((> cf9 0.0)
     (format t"~%fairly conclusive that the sensor has
failed"))
    ((> cf10 0.0)
     (format t"~%slightly conclusive that the sensor ha
s failed"))
    ((> cf11 0.0)
     (format t"~%strongly conclusive that the sensor is
 normal"))
    ((> cf12 0.0)
     (format t"~%fairly conclusive that the sensor is n
ormal"))
    ((> cf13 0.0)
     (format t"~%slightly conclusive that the sensor is
 normal"))
    ((> cf14 0.0)(format T"inconclusive")))
  (format T"~%because")
  (format T"~%there are ~d redundant sensors" (length r
edundant))
  (format T"~%and there are ~d groups of redundant sens
ors with
the same observed response to a system change" red-grou
ps2)
  (format T"~%and that the suspect sensor agrees with ~
d sensors
in the redundant list" (* red-agree2 (length redundant)
)))


;*****************************************************
*******************
(defun ss-change()
  (format T"~%The sensitivity test indicated the sensor
 was ")
  (cond((> cf26 0.0)(format T"failed"))
       (t(format T"normal")))
  (format T"~%The expected change was ~d ~d and
```

```
 the actual change was ~d ~d"
          (expected-change suspect)
          (sensorloop-units suspect)
          (sensorloop-change suspect)
          (sensorloop-units suspect)))


;*****************************************************
********************
(defun red-noise()
  (format T"~%the redundant analysis of the process noi
se is ")
  (cond
    ((> cf15 0.0)
     (format t"strongly conclusive that the sensor has
failed"))
    ((> cf16 0.0)
     (format t"fairly conclusive that the sensor has fa
iled"))
    ((> cf17 0.0)
     (format t"slightly conclusive that the sensor has
failed"))
    ((> cf18 0.0)
     (format t"~%strongly conclusive that the sensor is
 normal"))
    ((> cf19 0.0)
     (format t"fairly conclusive that the sensor is nor
mal"))
    ((> cf20 0.0)
     (format t"slightly conclusive that the sensor is n
ormal"))
    ((> cf21 0.0)(format T"inconclusive")))
  (format T"~%because")
  (format T"~%there are ~d redundant sensors" (length r
edundant))
  (format T"~%and there are ~d groups of sensors with s
imilar
response to process noise in the redundant list" red-gr
oups3)
  (format T"~%and that the suspect sensor agrees with ~
d sensors
in the redundant list" (* red-agree3 (length redundant)
)))


;*****************************************************
********************
(defun ss-noise()
  (format T"~%The expected fluctuation test indicat d t
he sensor was ")
  (cond((> cf28 0.0)(format T"failed"))
       (t(format T"normal")))
```

```
  (format T"~%The expected fluctuation was between ~d ~
d and ~d ~d and
 the actual fluctuation was ~d ~d"
          (sensorloop-maximum-fluctuation suspect)
          (sensorloop-units suspect)
          (sensorloop-minimum-fluctuation suspect)
          (sensorloop-units suspect)
          (sensorloop-fluctuation suspect)
          (sensorloop-units suspect)))


;*****************************************************
*******************
(defun overide()
  (cond((> cf32 0.0)
        (format t"~%~%~%the sensor appears to be operat
ional due to the
fact that redundant analysis of the three possible fail
ure modes indicated
 no evidence that the sensor was failed.")))
  (cond((and (> cf11 0.0)(> cf26 0.0))
             (format T"~%~%The sensitivity analysis sh
owed the sensor
 to respond incorrectly however its evaluation was over
ridden by the strong
evidence from a comparison of the change in the redunda
nt sensor's output
that the sensor was operational")
             (format T"~%The system specific informati
on has been
updated in order to account for this possible behavior
in the future~%~%")))
  (cond((and (> cf1 0.0)(> cf27 0.0))
             (format T"~%~%The limit checking test sho
wed the sensor
 to violate normal limits, however its evaluation was o
verridden by the strong
evidence from a comparison of the redundant sensor's av
erage output that
the sensor was operational")
             (format T"~%The system specific informati
on has been
 updated in order to account for this possible behavior
 in the future~%~%")))
  (cond((and (> cf18 0.0)(> cf28 0.0))
             (format T"~%~%The expected fluctuation te
st showed the
sensor to violate normal limits, however its evaluation
 was overridden by the
strong evidence from a comparison of the redundant sens
or's fluctuation that
the sensor was operational")
             (format T"~%The system specific informati
```

```
n has been
 updated in order to account for this possible behavior
 in the future~%~%")))
  )



;*****************************************************
*******************
;ROUTINES TO SET ANY NECCESARY VARIABLES
;*****************************************************
*******************
;INITIALIZE STARTING PARAMETERS
(setq current-power 0.0)
(setq suspect nil)



;*****************************************************
*******************
;INTIALIZE THE POSSIBLE GROUPS OF REDUNDANT SENSORS
(defun regroup()
  (setq group1 nil)
  (setq group2 nil)
  (setq group3 nil)
  (setq group4 nil)
  (setq group5 nil))

;RESET ALL VARIABLES PRIOR TO THE EVALUATION
(defun reload()
  (setq redundant nil)
  (setq location-list nil)
  (setq type-list nil)
  (setq red-groups1 0.0)
  (setq red-groups2 0.0)
  (setq red-groups3 0.0)
  (setq upstream nil)
  (setq same-location nil)
  (setq upstream-red nil)
  (setq downstream nil)
  (setq downstream-red nil)
  (setq number 0.0)
  (setq total 0.0)
  (setq red-value1 0.0)
  (setq red-value2 0.0)
  (setq red-value3 0.0)
  (setq red-agree1 0.0)
  (setq red-agree2 0.0)
  (setq red-agree3 0.0)
  (setq group1 nil)
  (setq group2 nil)
  (setq group3 nil)
  (setq group4 nil)
```

```
  (setq group5 nil)
  (setq up-agree 0.0)
  (setq down-agree 0.0))

(DEFUN RESeTCF()
  (SETQ CF1 0.0)(SETQ CF2 0.0)(SETQ CF3 0.0)(SETQ CF4 0
.0)(SETQ CF5 0.0)
  (SETQ CF6 0.0)(SETQ CF7 0.0)(SETQ CF8 0.0)(SETQ CF9 0
.0)(SETQ CF10 0.0)
  (SETQ CF11 0.0)(SETQ CF12 0.0)(SETQ CF13 0.0)(SETQ CF
14 0.0)(SETQ CF15 0.0)
  (SETQ CF16 0.0)(SETQ CF17 0.0)(SETQ CF18 0.0)(SETQ CF
19 0.0)(SETQ CF20 0.0)
  (setq cf21 0.0)(setq cf22 0.0)(setq cf23 0.0)(setq cf
24 0.0)(setq cf25 0.0)
  (setq cf26 0.0)(setq cf27 0.0)(setq cf28 0.0)(setq cf
29 0.0)(setq cf30 0.0)
  (setq cf31 0.0)(setq cf32 0.0)(setq cf33 0.0)(setq cf
34 0.0)(setq cf35 0.0)
  (setq facts nil)
  (setq temp1 nil)
  (setq asked whatnottoask)
  (setq rules rulescf)
  (SETQ HYPOTHESIS HYPOTHESISCF)
  )

;*****************************************************
*********************
;GENERAL PURPOSE ROUTINES
;*****************************************************
*********************
;*****************************************************
*********************
;CHECK FOR AGREEMENT OF SENSOR CHARACTEISTICS WITHIN TO
LERANCES
;*****************************************************
*********************
(defun eqtol(anum annum)
  (setq anum1 (float anum))
  (setq annum1 (float annum))
  (cond ((and (eql anum1 0.0)(eql annum1 0.0)) T)
        ((or (eql anum1 0.0)(eql annum1 0.0))
         (< (abs(- anum1 annum1)) (sensorloop-minimum-f
luctuation suspect)))
        ((or (< (abs anum1) 1)(< (abs annum1) 1))
         (< (abs (- anum1 annum1)) (* 10 (sensorloop-er
ror suspect))))
        (T(< (abs(/ (- anum1 annum1) anum1))
             (* 2.0 (sensorloop-error suspect))))))
;*****************************************************
*********************

;*****************************************************
```

```
********************
;FIND THE AVERAGE OF A LIST OF NUMBERS
(defun ave(alist)
  (cond ((NULL alist)(format T"~% there are no redundan
t sensors") nil)
        (t(/ (add alist) (length alist)))))


;*****************************************************
*********************
(defun add(alist)
  (cond ((NULL alist)(format T"~% there are no redundan
t sensors") nil)
        (T
            (prog(value)
              (setq value 0.0)
              loop
              (cond((null alist) (return value))
                   (t
                       (setq temp3 (car alist))
                       (setq value (+ (sensorloop-curre
nt-reading
                                                (eval temp3))
 value))
                       (setq alist (cdr alist))))
              (go loop)))))
;*****************************************************
*********************

;*****************************************************
*********************
;GENERAL PURPOSE ROUTINE TO APPEND TWO LISTS
(defun push1(l m)
  (setq l (append l m)))
;*****************************************************
*********************

;*****************************************************
*********************
;GENERAL ROUTINE TO VERIFY ASSERTIONS
(DEFUN EXAM(FACT)
  (eval fact))
;*****************************************************
*********************

;*****************************************************
*********************
(DEFUN MEMLIST(X Y)
  (COND ((NULL Y) NIL)
        ((EQUAL X (CAR Y)) T)
        (T (MEMLIST X (CDR Y)))))
;*****************************************************
*********************
```

```
(SETQ HYPOTHESISCf'
      (
       ((redundant analysis of average output
                   is strongly conclusive that the sens
or has failed)(cf1))
       ((redundant analysis of average output
                   is fairly conclusive that the sensor
 has failed)(cf2))
       ((redundant analysis of average output
                   is slighty conclusive that the senso
r has failed)(cf3))
       ((redundant analysis of average output
                   is strongly conclusive that the sens
or is normal)(cf4))
       ((redundant analysis of average output
                   is fairly conclusive that the sensor
 is normal)(cf5))
       ((redundant analysis of average output
                   is slighty conclusive that the senso
r is normal)(cf6))
       ((redundant analysis of average output is inconc
lusive)(cf7))
       ((redundant analysis of output change
                   is strongly conclusive that the sens
or has failed)(cf8))
       ((redundant analysis of output change
                   is fairly conclusive that the sensor
 has failed)(cf9))
       ((redundant analysis of output change
                   is slighty conclusive that the senso
r has failed)(cf10))
       ((redundant analysis of output change
                   is strongly conclusive that the sens
or is normal)(cf11))
       ((redundant analysis of output change
                   is fairly conclusive that the sensor
 is normal)(cf12))
       ((redundant analysis of output change
                   is slighty conclusive that the senso
r is normal)(cf13))
       ((redundant analysis of output change is inconcl
usive)(cf14))
       ((redundant analysis of average fluctuation
                   is strongly conclusive that the sens
or has failed)(cf15))
       ((redundant analysis of average fluctuation
                   is fairly conclusive that the sensor
 has failed)(cf16))
       ((redundant analysis of average fluctuation
                   is slighty conclusive that the senso
r has failed)(cf17))
       ((redundant analysis of average fluctuation
                   is strongly conclusive that the sens
```

```
(SETQ HYPOTHESISCf'
      (
       ((redundant analysis of average output
                   is strongly conclusive that the sens
or has failed)(cf1))
       ((redundant analysis of average output
                   is fairly conclusive that the sensor
 has failed)(cf2))
       ((redundant analysis of average output
                   is slighty conclusive that the senso
r has failed)(cf3))
       ((redundant analysis of average output
                   is strongly conclusive that the sens
or is normal)(cf4))
       ((redundant analysis of average output
                   is fairly conclusive that the sensor
 is normal)(cf5))
       ((redundant analysis of average output
                   is slighty conclusive that the senso
r is normal)(cf6))
       ((redundant analysis of average output is inconc
lusive)(cf7))
       ((redundant analysis of output change
                   is strongly conclusive that the sens
or has failed)(cf8))
       ((redundant analysis of output change
                   is fairly conclusive that the sensor
 has failed)(cf9))
       ((redundant analysis of output change
                   is slighty conclusive that the senso
r has failed)(cf10))
       ((redundant analysis of output change
                   is strongly conclusive that the sens
or is normal)(cf11))
       ((redundant analysis of output change
                   is fairly conclusive that the sensor
 is normal)(cf12))
       ((redundant analysis of output change
                   is slighty conclusive that the senso
r is normal)(cf13))
       ((redundant analysis of output change is inconcl
usive)(cf14))
       ((redundant analysis of average fluctuation
                   is strongly conclusive that the sens
or has failed)(cf15))
       ((redundant analysis of average fluctuation
                   is fairly conclusive that the sensor
 has failed)(cf16))
       ((redundant analysis of average fluctuation
                   is slighty conclusive that the senso
r has failed)(cf17))
       ((redundant analysis of average fluctuation
                   is strongly conclusive that the sens
```

```
r is normal)(cf18))
       ((redundant analysi  of average fluctuation
                    is fair y conclusive that the sensor
 is normal)(cf19))
       ((redundant analys s of average fluctuation
                    is sl: ghty conclusive that the senso
r is normal)(cf20))
       ((redundant anal sis of average fluctuation is i
nconclusive)(cf21))
       ((sensitivity of sensor to system change incorre
ct)(cf26))
       ((sensor output outside of normal limits)(cf27))
       ((sensor output fluctuation is outside normal li
mits)(cf28))
       ((sensor faile l due to incorrect average output)
(cf29))
       ((sensor fail ed due to incorrect response to a s
ystem change)(cf30)
       ((sensor fai ed due to incorrect response to pro
cess noise)(cf31))
       ((sensor op rational)(cf32))
       ))

(SETQ poss'
      (
       ((redunda t analysis of average output
                    is strongly conclusive that the sens
or has failed)( f1))
       ((reduno nt analysis of average output
                    is fairly conclusive that the sensor
 has failed)(c 2))
       ((redur lant analysis of average output
                    is slighty conclusive that the senso
r has failed) cf3))
       ((red dant analysis of average output
                    is strongly conclusive that the sens
or is norma )(cf4))
       ((re undant analysis of average output
                    is fairly conclusive that the sensor
 is normal (cf5))
       ((r dundant analysis of average output
                    is slighty conclusive that the senso
r is norm l)(cf6))
       (( edundant analysis of average output is inconc
lusive)(c 7))
       ( redundant analysis of output change
                    is strongly conclusive that the sens
or has ailed)(cf8))
        (redundant analysis of output change
                    is fairly conclusive that the sensor
 has f iled)(cf9))
       ((redundant analysis of output change
                    is slighty conclusive that the senso
r has failed)(cf10))
```

```
       ((redundant analysis of output change
                    is strongly conclusive that the sens
or is normal)(cf11))
       ((redundant analysis of output change
                    is fairly conclusive that the sensor
 is normal)(cf12))
       ((redundant analysis of output change
                    is slighty conclusive that the senso
r is normal)(cf13))
       ((redundant analysis of output change is inconcl
usive)(cf14))
       ((redundant analysis of average fluctuation
                    is strongly conclusive that the sens
or has failed)(cf15))
       ((redundant analysis of average fluctuation
                    is fairly conclusive that the sensor
 has failed)(cf16))
       ((redundant analysis of average fluctuation
                    is slighty conclusive that the senso
r has failed)(cf17))
       ((redundant analysis of average fluctuation
                    is strongly conclusive that the sens
or is normal)(cf18))
       ((redundant analysis of average fluctuation
                    is fairly conclusive that the sensor
 is normal)(cf19))
       ((redundant analysis of average fluctuation
                    is slighty conclusive that the senso
r is normal)(cf20))
       ((redundant analysis of average fluctuation is i
nconclusive)(cf21))
       ((sensitivity of sensor to system change incorre
ct)(cf26))
       ((sensor output outside of normal limits)(cf27))
       ((sensor output fluctuation is outside normal li
mits)(cf28))
       ((sensor failed due to incorrect average output)
(cf29))
       ((sensor failed due to incorrect response to a s
ystem change)(cf30))
       ((sensor failed due to incorrect response to pro
cess noise)(cf31))
       ((sensor operational)(cf32))
       ))


(SETQ whatnottoask'
      (
       (redundant analysis of average output
                    is strongly conclusive that the senso
r is normal)
       (redundant analysis of average output
                    is strongly conclusive that the senso
r has failed)
```

```
       (redundant analysis of average output
                  is fairly conclusive that the sensor
has failed)
       (redundant analysis of average output
                  is slighty conclusive that the sensor
 has failed)
       (redundant analysis of average output
                  is fairly conclusive that the sensor
is normal)
       (redundant analysis of average output
                  is slighty conclusive that the sensor
 is normal)
       (redundant analysis of average output is inconcl
usive)
       (redundant analysis of output change
                  is strongly conclusive that the senso
r is normal)
       (redundant analysis of output change
                  is strongly conclusive that the senso
r has failed)
       (redundant analysis of output change
                  is fairly conclusive that the sensor
has failed)
       (redundant analysis of output change
                  is slighty conclusive that the sensor
 has failed)
       (redundant analysis of output change
                  is fairly conclusive that the sensor
is normal)
       (redundant analysis of output change
                  is slighty conclusive that the sensor
 is normal)
       (redundant analysis of output change is inconclu
sive)
       (redundant analysis of average fluctuation
                  is strongly conclusive that the senso
r is normal)
       (redundant analysis of average fluctuation
                  is strongly conclusive that the senso
r has failed)
       (redundant analysis of average fluctuation
                  is fairly conclusive that the sensor
has failed)
       (redundant analysis of average fluctuation
                  is slighty conclusive that the sensor
 has failed)
       (redundant analysis of average fluctuation
                  is fairly conclusive that the sensor
is normal)
       (redundant analysis of average fluctuation
                  is slighty conclusive that the sensor
 is normal)
       (redundant analysis of average fluctuation is in
conclusive)
```

```
       (sensitivity of sensor to system change incorrec
t)
       (sensor output outside of normal limits)
       (sensor output fluctuation is outside normal lim
its)
       (sensor failed due to incorrect average output)
       (sensor failed due to incorrect response to a sy
stem change)
       (sensor failed due to incorrect response to proc
ess noise)
       (sensor operational)
       ))

(setq rulescf
      '(
        (rule 1
              (if
                  (>= (length redundant) 2)
                  (eql red-agreel 1.0)
                  )
              (then
                  ((redundant analysis of average outpu
t
                                is strongly conclusive th
at the sensor
                                is normal)(.99))
                  ))
        (rule 2
              (if
                  (>= (length redundant) 5)
                  (eql red-agreel (float(/ 4 5)))
                  )
              (then
                  ((redundant analysis of average outpu
t
                                is strongly conclusive th
at the sensor
                                is normal)(.99))
                  ))
        (rule 3
              (if
                  (eq (length redundant) 5)
                  (eql red-agreel (float(/ 3 5)))
                  )
              (then
                  ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                  ))
        (rule 4
              (if
                  (= (length redundant) 5)
```

```
                    (eql red-agree1 (float(/ 2 5)))
                    (eq red-groups1 4)
                    )
                (then
                    ((redundant analysis of average outpu
t
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 5
                (if
                    (= (length redundant) 4)
                    (eql red-agree1 (float(/ 2 4)))
                    (eql red-groups1 3)
                    )
                (then
                    ((redundant analysis of average outpu
t
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 6
                (if
                    (eq (length redundant) 4)
                    (eql red-agree1 (float(/ 3 4)))
                    (eq red-groups1 2)
                    )
                (then
                    ((redundant analysis of average outpu
t
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 7
                (if
                    (= (length redundant) 3)
                    (eql red-agree1 (float(/ 2 3)))
                    )
                (then
                    ((redundant analysis of average outpu
t
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 8
                (if
                    (= (length redundant) 1)
                    (eql red-agree1 (float(/ 1 1)))
                    )
                (then
```

```
                    ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                    ))
            (rule 9
                  (if
                      (= (length redundant) 5)
                      (eql red-agree1 (float(/ 3 5)))
                      (eq red-groups1 2)
                      )
                  (then
                      ((redundant analysis of average outpu
t
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                      ))
            (rule 10
                  (if
                      (= (lenght redundant) 4)
                      (eql red-agree1 (float(/ 2 4)))
                      (eq red-groups1 2)
                      )
                  (then
                      ((redundant analysis of average outpu
t
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                      ))
            (rule 11
                  (if
                      (eql red-agree1 (float(/ 1 (length re
dundant))))
                      (eq red-groups1 (length redundant))
                      )
                  (then
                      ((redundant analysis of average outpu
t
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                      ))
            (rule 12
                  (if
                      (eq (length redundant) red-groups1)
                      (eql red-agree1 0.0)
                      )
                  (then
                      ((redundant analysis of average outpu
t is inconclusive)(.99))
                      ))
```

```
        (rule 13
              (if
                  (eql (float(/ (length redundant) 2))
                       (float(* (length redundant) red-
agreel)))
                  )
              (then
                  ((redundant analysis of average outpu
t is inconclusive)(.99))
                  ))
        (rule 14
              (if
                  (eq (length redundant) 4)
                  (eq red-groups1 2)
                  (eql red-agree1 (float(/ 1 4)))
                  )
              (then
                  ((redundant analysis of average outpu
t
                              is slightly conclusive th
at the sensor has
                              failed)(.99))
                  ))
        (rule 15
              (if
                  (eq (length redundant) 4)
                  (eq red-groups1 3)
                  (eql red-agree1 (float(/ 0 4)))
                  )
              (then
                  ((redundant analysis of average outpu
t
                              is slightly conclusive th
at the sensor has
                              failed)(.99))
                  ))
        (rule 16
              (if
                  (eq (length redundant) 5)
                  (eq red-groups1 4)
                  (eql red-agree1 (float(/ 0 4)))
                  )
              (then
                  ((redundant analysis of average outpu
t
                              is slightly conclusive th
at the sensor has
                              failed)(.99))
                  ))
        (rule 17
              (if
                  (eq (length redundant) 5)
                  (eq red-groups1 2)
                  (eql red-agree1 (float(/ 1 5)))
```

```
                )
            (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.9^))
                ))
        (rule 18
            (if
                (eq (length redundant) 5)
                (eq red-groups1 3)
                (eql red-agree1 (float(/ 1 5)))
                )
            (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 19
            (if
                (eq (length redundant) 5)
                (eq red-groups1 2)
                (eql red-agree1 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 20
            (if
                (eq (length redundant) 5)
                (eq red-groups1 3)
                (eql red-agree1 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 21
            (if
                (eq (length redundant) 4)
                (eq red-groups1 2)
                (eql red-agree1 (float(/ 0 4)))
                )
            (then
```

```
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 22
              (if
                (eq (length redundant) 3)
                (eq red-groups1 2)
                (eql red-agree1 (float(/ 0 5)))
                )
              (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 23
              (if
                (eq (length redundant) 2)
                (eq red-groups1 1)
                (eql red-agree1 (float(/ 0 5)))
                )
              (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 24
              (if
                (eq (length redundant) 5)
                (eq red-groups1 2)
                (eql red-agree1 (float(/ 1 5)))
                )
              (then
                ((redundant analysis of average outpu
t
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 25
              (if
                (>= (length redundant) 3)
                (eq red-groups1 1)
                (eq red-agree1 (/ 0 5))
                )
              (then
                ((redundant analysis of average outpu
t
```

```
                                is strongly conclusive th
at the sensor has
                                failed)(.99))
                    ))
            (rule 26
                  (if
                      (>= (length redundant) 2)
                      (eql red-agree2 1.0)
                      )
                  (then
                      ((redundant analysis of output change
                                is strongly conclusive th
at the sensor
                                is normal)(.99))
                     ))
            (rule 27
                  (if
                      (>= (length redundant) 5)
                      (eql red-agree2 (float(/ 4 5)))
                      )
                  (then
                      ((redundant analysis of output change
                                is strongly conclusive th
at the sensor
                                is normal)(.99))
           ))
            (rule 28
                  (if
                      (eq (length redundant) 5)
                      (eql red-agree2 (float(/ 3 5)))
                      )
                  (then
                      ((redundant analysis of output change

                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                      ))
            (rule 29
                  (if
                      (= (length redundant) 5)
                      (eql red-agree2 (float(/ 2 5)))
                      (eq red-groups2 4)
                      )
                  (then
                      ((redundant analysis of output change

                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                      ))
            (rule 30
                  (if
                      (= (length redundant) 4)
```

```
                (eql red-agree2 (float(/ 2 4)))
                (eq red-groups2 3)
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor
                                is normal)(.99))
                ))
        (rule 31
            (if
                (= (length redundant) 4)
                (eql red-agree2 (float(/ 3 4)))
                (eq red-groups2 2)
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor
                                is normal)(.99))
                ))
        (rule 32
            (if
                (= (length redundant) 3)
                (eql red-agree2 (float(/ 2 3)))
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor
                                is normal)(.99))
                ))
        (rule 33
            (if
                (= (length redundant) 1)
                (eql red-agree2 (float(/ 1 1)))
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor
                                is normal)(.99))
                ))
        (rule 34
            (if
                (= (length redundant) 5)
                (eql red-agree2 (float(/ 3 5)))
                (eq red-groups2 2)
                )
            (then
```

```
                ((redundant analysis of output change
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                ))
        (rule 35
              (if
                (= (lenght redundant) 4)
                (eql red-agree2 (float(/ 2 4)))
                (eq red-groups2 2)
                )
              (then
                ((redundant analysis of output change
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                ))
        (rule 36
              (if
                (eql red-agree2 (float(/ 1 (length re
dundant))))
                (eq red-groups2 (length redundant))
                )
              (then
                ((redundant analysis of output change
                                is slightly conclusive th
at the sensor
                                is normal)(.99))
                ))
        (rule 37
              (if
                (eq (length redundant) red-groups2)
                (eql red-agree2 0.0)
                )
              (then
                ((redundant analysis of output change
 is inconclusive)(.99))
                ))
        (rule 38
              (if
                (eql (float(/ (length redundant) 2))
                     (float(* (length redundant) red-
agree2))))
              (then
                ((redundant analysis of output change
 is inconclusive)(.99))
                ))
        (rule 39
              (if
                (eq (length redundant) 4)
                (eq red-groups2 2)
```

```
                (eql red-agree2 (float(/ 1 4)))
                )
            (then
                ((redundant analysis of output change
                            is slightly conclusive th
at the sensor has
                            failed)(.99))
                ))
        (rule 40
            (if
                (eq (length redundant) 4)
                (eq red-groups2 3)
                (eql red-agree2 (float(/ 0 4)))
                )
            (then
                ((redundant analysis of output change
                            is slightly conclusive th
at the sensor has
                            failed)(.99))
                ))
        (rule 41
            (if
                (eq (length redundant) 5)
                (eq red-groups2 4)
                (eql red-agree2 (float(/ 0 4)))<
                )
            (then
                ((redundant analysis of output change
                            is slightly conclusive th
at the sensor has
                            failed)(.99))
                ))
        (rule 42
            (if
                (eq (length redundant) 5)
                (eq red-groups2 2)
                (eql red-agree2 (float(/ 1 5)))
                )
            (then
                ((redundant analysis of output change
                            is fairly conclusive that
 the sensor has
                            failed)(.99))
                ))
        (rule 43
            (if
                (eq (length redundant) 5)
                (eq red-groups2 3)
                (eql red-agree2 (float(/ 1 5)))
                )
```

```
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor has
                                failed)(.99))
                ))
        (rule 44
            (if
                (eq (length redundant) 5)
                (eq red-groups2 2)
                (eql red-agree2 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor has
                                failed)(.99))
                ))
        (rule 45
            (if
                (eq (length redundant) 5)
                (eq red-groups2 3)
                (eql red-agree2 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor has
                                failed)(.99))
                ))
        (rule 46
            (if
                (eq (length redundant) 4)
                (eq red-groups2 2)
                (eql red-agree2 (float(/ 0 4)))
                )
            (then
                ((redundant analysis of output change
                                is fairly conclusive that
the sensor has
                                failed)(.99))
                ))
        (rule 47
            (if
                (eq (length redundant) 3)
                (eq red-groups2 2)
                (eql red-agree2 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of output change
```

```
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 48
                  (if
                      (eq (length redundant) 2)
                      (eq red-groups2 1)
                      (eql red-agree2 (float(/ 0 5)))
                      )
                  (then
                      ((redundant analysis of output change

                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 49
                  (if
                      (eq (length redundant) 5)
                      (eq red-groups2 2)
                      (eql red-agree2 (float(/ 1 5)))
                      )
                  (then
                      ((redundant analysis of output change

                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 50
                  (if
                      (>= (length redundant) 3)
                      (eq red-groups2 1)
                      (eql red-agree2 (float(/ 0 5)))
                      )
                  (then
                      ((redundant analysis of output change

                                is strongly conclusive th
at the sensor has
                                failed)(.99))
                    ))
            (rule 51
                  (if
                      (>= (length redundant) 2)
                      (eql red-agree3 1.0)
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                is strongly conclusive th /
at the sensor
```

```
                                    is normal)(.99))
                    ))
            (rule 52
                  (if
                      (>= (length redundant) 5)
                      (eql red-agree3 (float(/ 4 5)))
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is strongly conclusive th
at the sensor
                                    is normal)(.99))
                    ))
            (rule 53
                  (if
                      (eq (length redundant) 5)
                      (eql red-agree3 (float(/ 3 5)))
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 54
                  (if
                      (= (length redundant) 5)
                      (eql red-agree3 (float(/ 2 5)))
                      (eq red-groups3 4)
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 55
                  (if
                      (= (length redundant) 4)
                      (eql red-agree3 (float(/ 2 4)))
                      (eq red-groups3 3)
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is fairly conclusive that
 the sensor
                                    is normal)(.99))
                    ))
            (rule 56
                  (if
```

```
                (= (length redundant) 4)
                (eql red-agree3 (float(/ 3 4)))
                (eq red-groups3 2)
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                ))
        (rule 57
            (if
                (= (length redundant) 3)
                (eql red-agree3 (float(/ 2 3)))
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                ))
        (rule 58
            (if
                (= (length redundant) 1)
                (eql red-agree3 (float(/ 1 1)))
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                ))
        (rule 58a
            (if
                (= (length redundant) 2)
                (eql red-agree3 0.5))
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor
                                is normal)(.99))
                ))
        (rule 59
            (if
                (= (length redundant) 5)
                (eql red-agree3 (float(/ 3 5)))
                (eq red-groups3 2)
                )
            (then
                ((redundant analysis of average fluct
```

```
ation
                                    is slightly conclusive th
at the sensor
                                    is normal)(.99))
                    ))
            (rule 60
                  (if
                      (= (lenght redundant) 4)
                      (eql red-agree3 (float(/ 2 4)))
                      (eq red-groups3 2)
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is slightly conclusive th
at the sensor
                                    is normal)(.99))
                      ))
            (rule 61
                  (if
                      (eql red-agree3 (float(/ 1 (length re
dundant))))
                      (eq red-groups3 (length redundant))
                      )
                  (then
                      ((redundant analysis of average fluct
uation
                                    is slightly conclusive th
at the sensor
                                    is normal)(.99))
                      ))
            (rule 62
                  (if
                      (eq (length redundant) red-groups3)
                      (eql red-agree3 0.0)
                      )
                  (then
                      ((redundant analysis of average fluct
uation is
                                    inconclusive)(.99))
                      ))
            (rule 63
                  (if
                      (> (length redundant) 2)
                      (eql (float(/ (length redundant) 2))
                           (float(* (length redundant) red-
agree3)))
                      )
                  (then
                      ((redundant analysis of average fluct
uation is
                                    inconclusive)(.99))
                      ))
            (rule 64
```

```
                (if
                    (eq (length redundant) 4)
                    (eq red-groups3 2)
                    (eql red-agree3 (float(/ 1 4)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is slightly conclusive th
at the sensor has
                                failed)(.99))
                    ))
        (rule 65
                (if
                    (eq (length redundant) 4)
                    (eq red-groups3 3)
                    (eql red-agree3 (float(/ 0 4)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is slightly conclusive th
at the sensor has
                                failed)(.99))
                    ))
        (rule 66
                (if
                    (eq (length redundant) 5)
                    (eq red-groups3 4)
                    (eql red-agree3 (float(/ 0 4)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is slightly conclusive th
at the sensor has
                                failed)(.99))
                    ))
        (rule 67
                (if
                    (eq (length redundant) 5)
                    (eq red-groups3 2)
                    (eql red-agree3 (float(/ 1 5)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
        (rule 68
                (if
                    (eq (length redundant) 5)
```

```
                    (eq red-groups3 3)
                    (eql red-agree3 (float(/ 1 5)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 69
                (if
                    (eq (length redundant) 5)
                    (eq red-groups3 2)
                    (eql red-agree3 (float(/ 0 5)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 70
                (if
                    (eq (length redundant) 5)
                    (eq red-groups3 3)
                    (eql red-agree3 (float(/ 0 5)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 71
                (if
                    (eq (length redundant) 4)
                    (eq red-groups3 2)
                    (eql red-agree3 (float(/ 0 4)))
                    )
                (then
                    ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                    ))
            (rule 72
                (if
                    (eq (length redundant) 3)
                    (eq red-groups3 2)
                    (eql red-agree3 (float(/ 0 5)))
```

```
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 73
            (if
                (eq (length redundant) 2)
                (eq red-groups3 1)
                (eql red-agree3 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 74
            (if
                (eq (length redundant) 5)
                (eq red-groups3 2)
                (eql red-agree3 (float(/ 1 5)))
                )
            (then
                ((redundant analysis of average fluct
uation
                                is fairly conclusive that
 the sensor has
                                failed)(.99))
                ))
        (rule 75
            (if
                (>= (length redundant) 3)
                (eq red-groups3 1)
                (eql red-agree3 (float(/ 0 5)))
                )
            (then
                ((redundant analysis of average fluct
uation
                                is strongly conclusive th
at the sensor has
                                failed)(.99))
                ))
        (rule 79
            (if
                (not(sensitivity-check suspect)))
            (then
                ((sensitivity of sensor to system cha
nge incorrect)(.99))
                ))
```

```
        (rule 80
              (if
                  (not(limit-check suspect)))
              (then
                  ((sensor output outside of normal lim
its)(.99))
                  ))
        (rule 81
              (if
                  (< (sensorloop-fluctuation suspect)
                     (sensorloop-minimum-fluctuation su
spect)))
              (then
                  ((sensor output fluctuation is outsid
e normal limits)(.99))
                  ))
        (rule 81.a
              (if
                  (> (sensorloop-fluctuation suspect)
                     (sensorloop-maximum-fluctuation su
spect)))
              (then
                  ((sensor output fluctuation is outsid
e normal limits)(.99))
                  ))
        (rule 82
              (if
                  (equal cf1 .99))
              (then
                  ((sensor failed due to incorrect aver
age output)(.99))
                  ))
        (rule 83
              (if
                  (equal cf2 .99)
                  (equal cf27 .99))
              (then
                  ((sensor failed due to incorrect aver
age output)(.85))
                  ))
        (rule 84
              (if
                  (equal cf2 .99)
                  (equal cf27 0.0))
              (then
                  ((sensor failed due to incorrect aver
age output)(.75))
                  ))
        (rule 85
              (if
                  (equal cf3 .99)
                  (equal cf27 .99))
              (then
                  ((sensor failed due to incorrect aver
```

```
ge output)(.75))
                    ))
        (rule 86
              (if
                  (equal cf3 .99)
                  (equal cf27 0.0))
              (then
                  ((sensor failed due to incorrect aver
age output)(.55))
                    ))
        (rule 87
              (if
                  (equal cf5 .99)
                  (equal cf27 .99))
              (then
                  ((sensor failed due to incorrect aver
age output)(.25))
                    ))
        (rule 88
              (if
                  (equal cf6 .99)
                  (equal cf27 .99))
              (then
                  ((sensor failed due to incorrect aver
age output)(.45))
                    ))
        (rule 89
              (if
                  (equal cf6 .99)
                  (equal cf27 0.0))
              (then
                  ((sensor failed due to incorrect aver
age output)(.25))
                    ))
        (rule 90
              (if
                  (equal cf7 .99)
                  (equal cf27 .99))
              (then
                  ((sensor failed due to incorrect aver
age output)(.55))
                    ))
        (rule 91
              (if
                  (equal cf15 .99))
              (then
                  ((sensor failed due to incorrect resp
onse to process noise)
                    (.99))
                  ))
        (rule 91
              (if
                  (equal cf16 .99)
                  (equal cf28 .99))
```

```
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.85))
                    ))
            (rule 92
                (if
                    (equal cf16 .99)
                    (equal cf28 0.0))
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.75))
                    ))
            (rule 93
                (if
                    (equal cf17 .99)
                    (equal cf28 .99))
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.75))
                    ))
            (rule 94
                (if
                    (equal cf17 .99)
                    (equal cf28 0.0))
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.55))
                    ))
            (rule 95
                (if
                    (equal cf19 .99)
                    (equal cf28 .99))
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.25))
                    ))
            (rule 96
                (if
                    (equal cf20 .99)
                    (equal cf28 .99))
                (then
                    ((sensor failed due to incorrect resp
onse to process noise)
                     (.45))
                    ))
            (rule 97
                (if
                    (equal cf20 .99)
                    (equal cf28 0.0))
```

```
            (then
                ((sensor failed due to incorrect resp
onse to process noise)
                 (.25))
                ))
        (rule 98
            (if
                (equal cf21 .99)
                (equal cf28 .99))
            (then
                ((sensor failed due to incorrect resp
onse to process noise)
                 (.55))
                ))
        (rule 100
            (if
                (equal cf8 .99))
            (then
                ((sensor failed due to incorrect resp
onse to a system change)
                 (.99))
                ))
        (rule 101
            (if
                (equal cf9 .99)
                (equal cf26 .99))
            (then
                ((sensor failed due to incorrect resp
onse to a system change)
                 (.85))
                ))
        (rule 102
            (if
                (equal cf9 .99)
                (equal cf26 0.0))
            (then
                ((sensor failed due to incorrect resp
onse to a system change)
                 (.75))
                ))
        (rule 103
            (if
                (equal cf10 .99)
                (equal cf26 .99))
            (then
                ((sensor failed due to incorrect resp
onse to a system change)
                 (.75))
                ))
        (rule 104
            (if
                (equal cf10 .99)
                (equal cf26 .00))
            (then
```

```
                ((sensor failed due to incorrect resp
onse to a system change)
                 (.55))
                ))
        (rule 107
              (if
                  (equal cf14 .99)
                  (equal cf26 .99))
              (then
                  ((sensor failed due to incorrect resp
onse to a system change)
                   (.45))
                  ))
        (rule 109
              (if
                  (equal cf13 .99)
                  (equal cf26 .99))
              (then
                  ((sensor failed due to incorrect resp
onse to a system change)
                   (.25))
                  ))
        (rule 110
              (if
                  (equal cf13 .99)
                  (equal cf26 .00))
              (then
                  ((sensor failed due to incorrect resp
onse to a system change)
                   (.10))
                  ))
        (rule 111
              (if
                  (equal cf12 .99)
                  (equal cf26 .99))
              (then
                  ((sensor failed due to incorrect resp
onse to a system change)
                   (.10))
                  ))
        (rule 113
              (if
                  (equal cf29 0.0)
                  (equal cf30 0.0)
                  (equal cf31 0.0))
              (then
                  ((sensor operational)(.99))
                  ))
        ))
```

```
#include <stdio.h>


float fluctuation(filename)
char filename[10];
{
float i,y,x,temp,temp2,sum;
FILE *fp;
fp = fopen(filename,"r");
x=0.0;
temp=0.0;
sum=0.0;
y=0.0;
i=0;
while(fscanf(fp,"%f", &temp)!=EOF) {
 sum =+ temp;
 i++;
 }
y = sum/i;
sum=0.0;
rewind(fp);
while(fscanf(fp,"%f", &temp)!= EOF){
 sum =+ (temp-y)*(temp-y);
 }
y = sum/(i-1);
fclose(fp);
return(y);
}
```

```
#include <stdio.h>

float average(filename)
char filename[10];
[
float i,y,x,temp,sum;
FILE *fp;
fp = fopen(filename,"r");
x=0.0;
temp=0.0;
sum=0.0;
y=0.0;
i=0;
while(fscanf(fp,"%f", &temp)!= EOF) {
 sum =+ temp;
 i++;
 }
y = sum/i;
fclose(fp);
return(y);
}
```

APPENDIX B

LISTING OF SAMPLE OUTPUT

```
Dribbling to SYS$USER5:[QUALLS.WORK]LIST.LSP;1
Lisp>
(expert)
"updating sensor readings...."

The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number

2

Enter the sensor you wish to have examined
fwsl517

the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
depentant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  1
 57.92 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
```

```
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  2
 57.93 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  3
 -0.009998 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
```

```
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  4
 56.5 59 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  5
 1.41 FEET and 0.156 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit
```

```
Please enter the appropriate number for the characteris
tic you need  6
 0.52345 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  7
 NIL
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  8
 0  unit change per unit change of independant paramete
r
the following are available characteristics of the sens
orloop
 1  current-reading
```

```
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  9
 DOWNCOMER
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  10
 LEVEL
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
```

```
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  11
 FEET
the following are available characteristics of the sens
orloop
 1  current-reading
 2  previous-reading
 3  change from previous evaluation
 4  upper and lower limits of normal operation for curr
ent power
 5  the limits of expected fluctuation of sensor output
 about the mean
 6  the fluctuation observed during last data acquistio
n
 7  which parameter this sensor is expected to change w
ith
 8  the expected change per unit for a change in the in
dependant variable
 9  sensor location
 10  sensor type
 11 units
 q quit

Please enter the appropriate number for the characteris
tic you need  q


The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
3
```

```
Enter the sensor you wish to have examined

If you wish to continue examining the same sensor type
 y

y

The redundant sensors are (FWSL518 FWSL519)

The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
4

Enter the sensor you wish to have examined

If you wish to continue examining the same sensor type
 y

y

The average output of the redundant sensors is 58.43

The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
```

```
umber
5

Enter the sensor you wish to have examined

If you wish to continue examining the same sensor type
 y

y

These sensors are located
at the same location as FWSL517 (FWSL517 FWSL518 FWSL51
9 FWSL501)


The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
6

 These are the type of sensors used to monitor the syst
em

(PRESSURE FLOW LEVEL TEMPERATURE)
 what type of sensors do you wish to have a list of
 enter type of sensor
flow

These sensors are all of the FLOW sensors (CVRCLF1 FWSF
510 FWSF511)


The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
```

```
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
7

the possible sensor locations are

(HOT-LEG COLD-LEG DOWNCOMER DOME STEAM-OULET FEEDWATER-
INLET)
From what location do you wish to have a list of sensor
s

 enter location
dome

These sensors are located
at the DOME (MSSP514 MSSP515 MSSP516)


The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
1

Enter the sensor you wish to have examined

If you wish to continue examining the same sensor type
 y

y

IT HAS BEEN DEDUCED BY USE OF RULE 1 that
(REDUNDANT ANALYSIS OF AVERAGE OUTPUT IS STRONGLY CONCL
USIVE THAT THE SENSOR IS NORMAL)
```

```
has confidence
(0.99)
IT HAS BEEN DEDUCED BY USE OF RULE 38 that
(REDUNDANT ANALYSIS OF OUTPUT CHANGE IS INCONCLUSIVE)
has confidence
(0.99)
IT HAS BEEN DEDUCED BY USE OF RULE 51 that
(REDUNDANT ANALYSIS OF AVERAGE FLUCTUATION IS STRONGLY
CONCLUSIVE THAT THE SENSOR IS NORMAL)
has confidence
(0.99)
IT HAS BEEN DEDUCED BY USE OF RULE 113 that
(SENSOR OPERATIONAL)
has confidence
(0.99)
NIL
 CONCLUSIONS IN DECENDING PRIORITY
**********************************************************
*********
The sensor is operational


**********************************************************
*********
Do you wish to see the facts and results which lead to
the
            conclusions of the evaluation ? (y/n)
 y
the redundant analysis of the average output is
strongly conclusive that the sensor is normal because
there are 2 redundant sensors
and there are 1 groups with the same average
  output within the redundant list
and that the suspect sensor agrees with 2.0 sensors
 in the redundant list
The limit checking test indicated the sensor was normal
The expected limits of operation
are 59 and 56.5 and the sensor had an output of 57.92 F
EET
the redundant analysis of the output change is inconclu
sive because
there are 2 redundant sensors
and there are 2 groups of redundant sensors with
the same observed response to a system change
and that the suspect sensor agrees with 1.0 sensors
in the redundant list
The sensitivity test indicated the sensor was normal
The expected change was 0.0 FEET and
 the actual change was -0.009998 FEET
the redundant analysis of the process noise is
strongly conclusive that the sensor is normal because
there are 2 redundant sensors
and there are 1 groups of sensors with similar
response to process noise in the redundant list
```

```
and that the suspect sensor agrees with 2.0 sensors
in the redundant list
The expected fluctuation test indicated the sensor was
normal
The expected fluctuation was between 1.41 FEET and 0.15
6 FEET and
 the actual fluctuation was 0.52345 FEET


the sensor appears to be operational due to the
fact that redundant analysis of the three possible fail
ure modes indicated
 no evidence that the sensor was failed.
Please enter   y   when ready to resume the consultatio
n

y


The following is a list of sensors currently available
for analysis
(MSSP514 MSSP515 MSSP516 CVRCLF1 RCST413A RCST413B FWSF
510 FWSF511 FWSL517
 FWSL518 FWSL519 FWSL501)
The following is a list of possible procedures
 1  perform an evaluation of a sensor's performance
 2  obtain the current value a sensor parameter
 3  obtain a list of sensors redundant to any other
 4  obtain the average of the redundant sensors output
 5  obtain a list of sensors at the same location as an
y other
 6  obtain a list of all sensors of a specific type
 7  obtain a list of all sensors at a specific location
 Q  quit
Please specify your request by entering the appropiate
number
q
T
Lisp> (dribble)
```

```
(setq locations '(hot-leg cold-leg downcomer dome steam
-oulet feedwater-inlet))
(setq sensor-types '(pressure flow level temperature))
(setq hot-leg 1)
(setq cold-leg 2)
(setq feedwater-inlet 6)
(setq downcomer 7)
(setq dome 8)
(setq steam-outlet 9)
(setq sensorloops '(mssp514 mssp515 mssp516 cvrclf1 rcs
t413a rcst413b
                          fwsf510 fwsf511 fwsl517 fws
l518 fwsl519 fwsl501))

(setq mssp514 (make-sensorloop
                  :name 'mssp514
                  :kind 'pressure
                  :location 'dome
                  :filename '"mssp514.dat"
                  :sensitivity -.618
                  :sensitive-to 'power
                  :change 0
                  :maximum-fluctuation 11.97
                  :minimum-fluctuation 1.33
                  :limit1 0
                  :upper-limit1 1000
                  :lower-limit1 955
                  :upper-limit2 1000
                  :lower-limit2 955
                  :result-filename '"mssp514r.dat"
                  :current-reading 983.0509
                  :previous-reading 0
                  :units 'psia
                  :error .025
                  ))

(setq mssp515 (make-sensorloop
                  :name 'mssp515
                  :kind 'pressure
                  :location 'dome
                  :filename '"mssp515.dat"
                  :sensitivity -.618
                  :sensitive-to 'power
                  :change 0
                  :maximum-fluctuation 11.97
                  :minimum-fluctuation 1.33
                  :limit1 0
                  :upper-limit1 1000
                  :lower-limit1 955
                  :upper-limit2 1000
                  :lower-limit2 955
                  :result-filename '"mssp515r.dat"
                  :current-reading 986.625
                  :previous-reading 0
```

```
                    :units 'psia
                    :error .025
                    ))

(setq mssp516 (make-sensorloop
                    :name 'mssp516
                    :kind 'pressure
                    :location 'dome
                    :filename '"mssp516.dat"
                    :sensitivity -.618
                    :sensitive-to 'power
                    :change 0
                    :maximum-fluctuation 11.97
                    :minimum-fluctuation 1.33
                    :limit1 0
                    :upper-limit1 1000
                    :lower-limit1 955
                    :upper-limit2 1000
                    :lower-limit2 955
                    :result-filename '"mssp516r.dat"
                    :current-reading 986.1049
                    :previous-reading 0
                    :units 'psia
                    :error .025
                    ))

(setq cvrclf1 (make-sensorloop
                    :name 'cvrclf1
                    :kind 'flow
                    :location 'cold-leg
                    :filename '"cvrclf1.dat"
                    :sensitive-to 'power
                    :change -0.01
                    :sensitivity 0
                    :minimum-fluctuation .09
                    :maximum-fluctuation .84
                    :limit1 100
                    :upper-limit1 111.9
                    :lower-limit1 110.6
                    :upper-limit2 111.9
                    :lower-limit2 110.6
                    :result-filename '"cvrcl1r.dat"
                    :current-reading 111.344
                    :previous-reading 0
                    :units 'thousand-gal/min
                    :error .015
                    ))

(setq fwsl517 (make-sensorloop
                    :name 'fwsl517
                    :kind 'level
```

```
                    :location 'downcomer
                    :filename '"fwsl517.dat"
                    :sensitive-to nil
                    :sensitivity 0
                    :change 0
                    :minimum-fluctuation .156
                    :maximum-fluctuation 1.41
                    :limit1 70
                    :upper-limit1 50
                    :lower-limit1 48
                    :upper-limit2 59
                    :lower-limit2 56.5
                    :result-filename '"fwsl517r.dat"
                    :current-reading 57.93
                    :previous-reading 0
                    :units 'feet
                    :error .015
                    ))

(setq fwsl518 (make-sensorloop
                    :name 'fwsl518
                    :kind 'level
                    :location 'downcomer
                    :filename '"fwsl518.dat"
                    :sensitive-to nil
                    :sensitivity 0
                    :change 0
                    :minimum-fluctuation .156
                    :maximum-fluctuation 1.41
                    :limit1 70
                    :upper-limit1 50
                    :lower-limit1 48
                    :upper-limit2 59
                    :lower-limit2 56.5
                    :result-filename '"fwsl518r.dat"
                    :current-reading 58.22099
                    :previous-reading 0
                    :units 'feet
                    :error .015
                    ))

(setq fwsl519 (make-sensorloop
                    :name 'fwsl519
                    :kind 'level
                    :location 'downcomer
                    :filename '"fwsl519.dat"
                    :sensitive-to nil
                    :sensitivity 0
                    :change 0
                    :minimum-fluctuation .156
                    :maximum-fluctuation 1.41
                    :limit1 70
```

```
                    :upper-limit1 50
                    :lower-limit1 48
                    :upper-limit2 59
                    :lower-limit2 56.5
                    :result-filename '"fwsl519r.dat"
                    :current-reading 58.32
                    :previous-reading 0
                    :units 'feet
                    :error .015
                    ))

(setq fwsl501 (make-sensorloop
                    :name 'fwsl501
                    :kind 'level-wide-range
                    :location 'downcomer
                    :filename '"fwsl501.dat"
                    :sensitive-to nil
                    :sensitivity 0
                    :change 0
                    :minimum-fluctuation .28
                    :maximum-fluctuation 2.52
                    :limit1 70
                    :upper-limit1 52.0
                    :lower-limit1 48
                    :upper-limit2 63.0
                    :lower-limit2 56.5
                    :result-filename '"fwsl501r.dat"
                    :current-reading 62.115
                    :previous-reading 0
                    :units 'feet
                    :error .025
                    ))

(setq fwsf510 (make-sensorloop
                    :name 'fwsf510
                    :kind 'flow
                    :location 'steam-outlet
                    :filename '"fwsf510.dat"
                    :sensitive-to 'fwsf511
                    :sensitivity 1
                    :change 0
                    :minimum-fluctuation 12.2
                    :maximum-fluctuation 110.01
                    :limit1 100
                    :lower-limit1 1415
                    :upper-limit1 3744
                    :lower-limit2 1415
                    :upper-limit2 3744
                    :result-filename '"fwsf510r.dat"
                    :current-reading 2693.819
                    :previous-reading 0
                    :units 'lbm/sec
```

```
                :error .035
                ))

(setq fwsf511 (make-sensorloop
                :name 'fwsf511
                :kind 'flow
                :location 'feedwater-inlet
                :filename '"fwsf511.dat"
                :sensitive-to 'fwsf510
                :sensitivity 1
                :change 0
                :minimum-fluctuation 12.2
                :maximum-fluctuation 110.01
                :limit1 100
                :lower-limit1 1415
                :upper-limit1 3744
                :lower-limit2 1415
                :upper-limit2 3744
                :result-filename '"fwsf511r.dat"
                :current-reading 2696.816
                :previous-reading 0
                :units 'lbm/sec
                :error .005
                ))

(setq rcst413b (make-sensorloop
                :name 'rcst413b
                :kind 'temperature
                :location 'cold-leg
                :filename '"rcst413b.dat"
                :sensitive-to nil
                :sensitivity 0
                :change 0
                :minimum-fluctuation .38
                :maximum-fluctuation 3.45
                :limit1 100
                :upper-limit1 558
                :lower-limit1 552.5
                :upper-limit2 558
                :lower-limit2 552.5
                :result-filename '"rcst413br.dat"
                :current-reading 555.408
                :previous-reading 0
                :units 'deg_F
                :error .015
                ))

(setq rcst413a (make-sensorloop
                :name 'rcst413a
                :kind 'temperature
                :location 'hot-leg
```

```
:filename '"rcst413a.dat"
:sensitive-to 'power
:sensitivity .510
:change 0
:minimum-fluctuation .383
:maximum-fluctuation 3.45
:limit1 100
:upper-limit1 613
:lower-limit1 575
:upper-limit2 613
:lower-limit2 575
:result-filename '"rcst413ar.dat"
:current-reading 598.772
:previous-reading 0
:units 'deg_F
:error .015
))
```

VITA

Arthur Louis Qualls

Arthur Louis Qualls was born on September 12, 1963. He was born and raised in Memphis, Tennessee. After twelve years of wandering from school to school looking for some direction in his life, he settled in Knoxville, Tennessee and began his undergraduate studies at the University of Tennessee. After one more year of wandering, he wandered into the office of Dr. P.F. Pasqua, Head of the Nuclear Engineering Department, and there found the direction that he had longed for all those years.

About this time he wandered on to the lacrosse field, was fascinated by the odd sport and spent the next seven springs getting bumped and bruised in the pursuit of happiness, which was his inalienable right. Other pursuits of Mr. Qualls was a job as chief mechanic for a locally owned racing boat and lead guitarist in a locally unheard of rock and roll band.

After receiving his B.S. Degree with a major in Nuclear Engineering in spring of 1986, he decided he needed more direction and re-upped for another tour of duty. Upon the completion of the requirements for a M.S. with a major in Nuclear Engineering from the University of Tennessee at

Knoxville in June 1988, he was informed that he had to write a summary of his life story. That is what he is doing now.

Mr. Qualls plans to either pursue a career in Nuclear Engineering or professional hockey upon graduation.