A BEST ESTIMATE METHOD FOR THE DIAGNOSIS AND MITIGATION OF MULTIPLE-FAILURE TRANSIENTS IN NUCLEAR POWER PLANTS

A Thesis

by

ROBERT PAUL MARTIN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 1989

Major Subject: Nuclear Engineering

A BEST ESTIMATE METHOD FOR THE DIAGNOSIS AND MITIGATION OF MULTIPLE-FAILURE TRANSIENTS IN NUCLEAR POWER PLANTS

A Thesis

by

ROBERT PAUL MARTIN

Approved as to style and content by:

B. Nassersharif
(Chair of Committee)

K. L. Peddicord
(Member)

C. D. Boyle
(Member)

J. W. Poston
(Head of Department)

May 1989

ABSTRACT

A Best Estimate Method for the Diagnosis and Mitigation of Multiple-Failure Transients in Nuclear Power Plants. (May 1989)
Robert P. Martin, B. S., Texas A & M University
Chair of Advisory Committee: Dr. Bahram Nassersharif

This software was designed to diagnose major nuclear power plant transients in real-time by analyzing simulated sensor and plant thermal hydraulic information, typical of that available in a nuclear power plant control room and is able to provide emergency response operating procedures for mitigating transient scenarios. The purpose of this code is to analyze incoming data from a nuclear power plant (or from a power plant simulation), interpret the control system data over consecutive time increments, determine the transient sequence, allow the operator to add relevant information, return diagnosis and suggested remedy responses (taken from nuclear-plant emergency operating guidelines). This expert system specifically addresses issues of uncertainty management, multiple failure, real-time, best estimate diagnosis, information management, and mitigation procedures.

The "best-estimate" strategy presented here involves using a modification of assumption-based truth maintenance system (ATMS) theory to improve diagnosis. This method extends the ATMS technique by incorporating a means for ignoring minor conflicts in the knowledge base, known as confidence level assessment. This allows for a simpler, more efficient knowledge base, making real time diagnosis possible. This modification requires that as part of the assumption set (i.e., heuristic rules and procedures), weighing factors are assigned to each assumption, these weighing factors are manipulated by an inference engine in such a way as to accommodate for this problem. A few expert systems have been developed that apply a deviation of this idea, concentrating on applying a theory of probability. However, the work presented herein is based on a different method involving "confidence levels".

For this project a qualitative model of a nuclear power plant and the ATMS knowledge base of transient facts were developed as part of a software experiment demonstrating the software and the methods on which it is based. Four case studies were performed and evaluated.

ACKNOWLEDGEMENT

This work presented in this thesis could not have been completed without the help of a number of persons. I would like to thank my committee chairman, Dr. Bahram Nassersharif, for the encouragement, motivation, and direction without which this work would have not been possible. I also appreciate the cooperation of Mrs. Evelyn Mullen who provided certain literature essential to this work. I also wish to thank my family and colleagues whose constant support and understanding greatly helped in the completion of this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

Acronym	Name
AI	Artificial Intelligence
ATMS	Assumption-Based Truth Maintenance System
ATWS	Anticipated Transient Without Scram
CATALisp	Computer Aided Transient Analysis encoded in Lisp
ES	Expert System
FSAR	Final Safety Analysis Report
HPIS	High Pressure Injection System
LOCA	Loss-of-Coolant Accident
LOFW	Loss-of-Feedwater
LOSP	Loss of Offsite Power
NRC	Nuclear Regulatory Commission
PWR	Pressurized Water Reactor
RCS	Reactor Coolant System
RCP	Reactor Coolant Pump
SCS	Secondary Coolant System
SG	Steam Generator
SGTR	Steam Generator Tube Rupture
TMS	Truth Maintenance System
TRAC	Transient Reactor Analysis Code
WEROG	Westinghouse Emergency Response Operator Guidelines

CHAPTER I

INTRODUCTION

The purpose of this work was to develop an expert system (ES) using assumption-based truth maintenance theory capable of monitoring a "simulated" nuclear reactor facility. The ES would detect deviations from normal operation, diagnose the event, and propose possible mitigation procedures to resolve the problem. Such an expert system must address many issues that will affect performance.

Normal operation of a nuclear power plant is controlled by the plant trip and control system. When an emergency occurs, an accurate and thorough understanding of the state of the reactor is necessary to diagnose and mitigate the transient because consequences could be severe. For example, during the first few minutes of an event, as many as 100 annunciators can alarm in the control room. Reaction to this information overload and the mechanical failure of certain sensors can lead to a misinterpretation of the situation; consequently, a plant can be permanently damaged and safety of the public may be at risk.

For a proper evaluation of a potential accident, it is necessary to closely monitor the many components of a nuclear reactor system. Ideally, an operator would want to know the exact state of the reactor at all times. In cases of multiple-failure transients, diagnosis is more difficult because the existing operating guidelines do not sufficiently cover such cases. Often the failure of one component causes the failure of another or interferes with mitigation of the transient. An alternative to the current operating environment would be to introduce intelligent software to perform analysis on occurring plant conditions. A computer could be interfaced with the existing plant computer which collects plant information and could analyze many transients in real time. Analysis of plant control system data could then be correlated against operator emergency response guidelines and a corresponding mitigating response could be determined. In complex systems such as nuclear power plants allowances must also be made for lost or invalid sensor data.

---

*This thesis follows the journal style recommended by Nuclear Technology.

For complex systems such as nuclear power plants, operators are trained to deal with diagnosis and mitigation of a wide variety of event scenarios. During emergency situations, plant conditions can present a challenge to an operator's training experience. Information overload, multiple failure, and uncertainty can contribute additional problems in diagnosing and mitigating system faults.

During the operation of nuclear power plants (or their numerical simulation), much data is produced describing current conditions of every subsystem and components inherent to the plant. Information is generated in the form of thermal-hydraulic data, annunciator status, sensor data, and other data which would be available in a nuclear power plant control room. When a transient occurs, an operator experiences an abundance of information which can be difficult to assess. An experienced operator or analyst can diagnose a transient effectively by examining the proper gauges and annunciators that will give him the information essential for diagnosis and mitigation of the transient. Likewise, for an expert system designed to diagnose reactor transients, it must know what essential data to reference. The quantity of this information load is great even for the computer; therefore, this expert system is designed for sequential deduction, similar to the operator.[1]

Another strategy for maximizing the information from a minimum number of data sources is to concentrate on data that is unique to a transient and is most often available. By analyzing transient calculation data, such as that produced by TRAC,[2] an understanding of the essential data needed for diagnosis can be established. Since the expert system is designed to be real-time, it can increase the amount of information that can be learned from plant data by analyzing change over a length of time. Once the essential data sources are determined, the method of sequential deduction can be followed.

During multiple-failure reactor transients, the occurrence of a second transient can override the signatures that identify a first transient that may have been present for some time. Such cases pose a difficult problem for correctly identifying both transients. Strict appliance of the symptom oriented diagnostic methods currently used at nuclear power plants cannot be performed. An operator may have to rely only his intuition and experience to derive his best-estimate diagnosis of the event.

Sometimes failures can occur with instrumentation, resulting in lost or invalid information required by the operator. An operator must then make his decision based on a subset of the total information he needs for a thorough analysis. If the uncertainty

is great, he faces a similar problem as with multiple failure, strict application of established diagnostic method cannot be used.

Due to the likelihood of invalid or unavailable data, multiple failure, information overload, and the need for quick response to system failures, a significant amount of uncertainty exists for a nuclear power plant operator during a transient situation. The expert reactor operator can respond to this information from experience, research, learning, or intuition; however, these concepts are not programmable into a mechanistic strategy. By using expert system techniques, those concepts can be captured to a degree. An expert system, as defined in Putting Expert System into Practice,[3] is a computer system capable of simulating that element of a human specialist's knowledge and reasoning that can be formulated into knowledge chunks characterized by a set of facts and heuristic rules. Heuristic rules are rules of thumb accumulated by a human expert through intensive problem solving in the domain of a particular task. The main body of an expert system usually consists of an inference engine. The inference engine is that part of the code that evaluates rules that define the expert knowledge. For the uncertainty that exists in nuclear power plants, system evaluation requires a "best-estimate" reasoning from known as well as heuristic conditions. Because mechanistic algorithms are incapable of this form of reasoning, the application of expert system methods capable of performing this kind of analysis is necessary.

From a user standpoint, the main difference between "intelligent" and "number crunching" computer systems is in the datum that can be manipulated. Traditional "number crunching" oriented computers manipulate numbers by mechanistic algorithms. "Intelligent" computers have that added capability of being able to manipulate symbols (i.e., words, phrases, etc.). This ability has created new strategies for problem solving. With symbol manipulation integrated with traditional "number crunching", knowledge can be captured in words and phrases compiled into a qualitative model or heuristic rules. Predictions of state, as well as other information, can be inferred from these rules which describe action and function of specific components integrated in a system.

Emulation of the diagnostic strategies of an expert nuclear reactor operator requires an evaluation of methods that can be provided by reactor operator guidelines or plant final safety analysis reports, and to compile them into a qualitative model or statements of procedure that an operator would follow during a reactor transient situation. A model would include instruments and gauges found in the reactor control

room that the operator would monitor and basic knowledge of the system as a whole. Statements of procedure would be based on intuitive analysis of trends occurring during a particular transient.

All of this information is used by the operator in order to answer questions about the reactor's condition. Likewise to emulate such a process, the computer must transform such information into a symbolic form and match it against heuristic rules and a qualitative knowledge base describing the system. This would include a compilation of all the knowledge a reactor operator would rely on during abnormal conditions. For example, an expert reactor operator knows that if there is a loss-of-coolant-accident (LOCA), the primary pressure will decrease. The computer can emulate this by comparing the pressure at two different times, if there is a drop in pressure, it can label this condition as DECREASING PRESSURE on the PRIMARY COOLANT LINE. A rule can be developed as IF THE PRESSURE on the PRIMARY COOLANT LINE is DECREASING, then CHECK FOR LOCA.

The state-of-the-art in Artificial Intelligence (AI) and Expert System technology has matured to a degree that the potential development of a computer aided/automated diagnostic and transient mitigation system can be considered. Since traditional methods cannot handle complex systems efficiently, AI techniques provide a means to emulate an expert reactor operator rather than following mechanistic methods.

Assumption-based truth maintenance systems (ATMS) represent an aspect of this state-of-the-art technology directly applied to problem-solving. The ATMS is based on the manipulation of assumption sets. As a consequence it is possible to work effectively and efficiently with inconsistent information and to examine more than one point in the search space at one time, thus, ATMS is a problem-solving architecture in which all potential solutions are explored simultaneously.

The software has been entitled CATALisp for Computer Aided Transient Analysis encoded in Lisp.

A. Literature Review

The application of AI techniques in the form of expert systems to nuclear engineering problems has been examined by numerous other organizations and individuals; however, this research has been limited by the state-of-the-art in expert system technology at the time of the research. Since the AI state-of-the-art is rapidly

changing year-to-year, such research becomes outdated as new methods and better facilities become available; yet, the research is still significant to current projects, since they can provide the heuristic and qualitative knowledge that is inherent to expert system development as applied to nuclear engineering.

One of the first significant applications of expert system technology was REACTOR[4]. This program can be described as a rule-based expert system designed for simple analysis of nuclear reactors. This expert system queried the user for symptoms of failure in a nuclear power plant and provided a final diagnosis. A similar expert system for improving operator diagnostic ability has been developed by Wells and Underwood.[5,6] The U.S. Nuclear Regulatory Commission (NRC)[7] has under development an expert system called The Reactor Safety Assessment System. This system generates conclusions for assessed situations given only parametric values, known operator actions, and transient information in the data. At the University of Michigan a methodology for combining model- and rule-based algorithms has been developed specifically for the purpose of diagnosing off-normal events in nuclear power plants from thermal-hydraulic conditions in the plant in real-time.[8] Also, much research in developing a sensor validation system is being conducted at Ohio State University.[9,10] More recently, an expert system ICS-EXPERT[11] used probabilistic risk assessment and abnormal transient operating guidelines for real-time diagnosis as opposed to "snapshot" diagnosis at some arbitrary time.

At the Experimental Breeder Reactor-II (EBR-II) at Idaho National Engineering Laboratory two programs have been developed for analysis and diagnosis of that reactor. The System State Analyzer[12] and DISYS[13] are expert systems that receive real-time plant data and perform rule-base inference to discover potential problems in the cooling system of the EBR-II.

A number of other applications of expert systems to plant operations have been described. These include a program for probabilistic risk assessment,[14] operations analysis of the Savannah River reactors,[15] automated monitoring of plant performance for the Oak Ridge National Laboratory High Flux Intensity Reactor,[16] and refuelling assistance on the Fast Flux Test Facility.[17]

B. Methodology and Results

Development of this work requires investigation into issues of qualitative modelling and simulation, transient diagnosis and mitigation, and information management. Chapter II details qualitative modelling and simulation methods as they pertain to expert systems. Development of the strategies and issues relevant for transient diagnosis are explained in Chapter III and methods for transient mitigation are address in Chapter IV. Chapter V presents the software experiment performed for evaluating the expert system. A full description of the software and results from codifying expertise and from four simulation case studies are present in this chapter. Evaluation of the methods used and results obtained, along with general conclusions and discussion is provided in Chapter VI.

CHAPTER II

QUALITATIVE MODELLING AND SIMULATION

A. Qualitative Modelling

A model is an intellectual construct used for studying the behavior of a physical system.[18] Two types of models exist for describing the physical world, quantitative and qualitative. While the goal of both modelling schemes is to make predictions or to explain properties of systems with equally precise statements of fact, the methods are quite different. Quantitative models describe methodologies that incorporate the capabilities of precise measurements. Conversely, a qualitative model is not concerned with precision and quantification is not of prime importance.

A goal of qualitative physics is to develop systematic models that capture the breath and depth of human reasoning about the physical world. However, the main obstacles inhibiting such models are availability of the expert knowledge and computer limitations in speed and memory. These models must be capable of capturing several levels of detail, and a variety of different perspectives. Most qualitative models can be described as scenario models because they describe a specific system or situation. In most cases, a model must be carefully designed so that rules used by an inference engine may reference model information effectively.

There are both advantages and disadvantages to having a very large or very small qualitative model. A very large model can be designed to be very generalized; allowing for greater flexibility in analyzing events for different types of systems and greater ability to derive precise predictions about the model. However, these models require more effort to construct and more effort by the expert system to retrieve information. Small models, although they must sacrifice precision, are more easily handled by the expert system. However, every model deviates from the physical system in order to simplify it. Simplification is both legitimate and necessary, but such assumptions must be accounted for when interpreting the model's predictions.

A qualitative model of a nuclear power plant must adequately describe all relevant components; however, it may not be necessary to model every component. Groups of components working as a system can be modelled as a system, such as the High

Pressure Injection System (HPIS). The purpose of the HPIS is to supply coolant at high pressure to the core during unexpected depressurization of the primary coolant system. The HPIS can consist of a water holding tank, a pump, water lines, and a sparger; however it is not necessary to model each component. By isolating what component sets can be classified as subsystems, the size of the qualitative model is minimized. This system subset can be minimized further to account for only those components that supply necessary information for performing diagnosis of the system. This restricts any redundant information from the model.

To meet a requirement for near real time execution, the qualitative model used in this work has been structured to include only the most relevant and necessary information, required by CATALisp to identify the specific set of transients in its knowledge base, in an attempt to optimize the use of the model. The qualitative model used in this work can be found in Appendix C. A detailed description of this model is provided in Chapter VI.

B. Qualitative Simulation

Human reasoning is able to perform extremely well with vague, context-sensitive concepts, such as high, low, big, small, etc. This is intimately related to our ability to make decisions based on common sense knowledge. Computers, however, deal most effectively with specific numerical values for comparison and judgement. Although AI research has resulted in inexact or fuzzy logic theory, expert system technologies are currently limited in their ability to perform the kinds of qualitative simulations that humans spontaneously employ when reasoning and making decisions.

While the exact emulation of human reasoning is a topic of current research, good results have been acquired from attempts at approximating it. Simple nuclear power plant analysis concepts can be derived from simulation data to be used by rules and an inference engine. Knowledge of normal operational conditions included in the qualitative model can be compared to the "current" conditions to establish qualitative measures such as high, low, normal, on or off. For a real time system additional qualitative knowledge can be captured by using a first order approximation to parameter derivatives over a transient time step, such as decreasing, increasing, or stable. These concepts are used to update the status of the model, specifically the conditions in a component or system (i.e., operational status, pressure, temperature, etc.). In this

way all numerical data from a simulation is converted into a symbolic form. As an example, if pressure in the reactor coolant system drops from 2250 psia to 2235 psia in one second, the program can convert this data to a pressure that is "low" (with respect to the normal pressure of 2250 psia), a pressure derivative that is "decreasing", and a pressure rate that is "quickly." Concepts such as "quickly" are more abstract than "low" or "decreasing," therefore they are usually defined for specific purposes, such as distinguishing between two different results (e.g., a pressure rate of "quickly" might suggest a transient condition that is more severe than the condition of a pressure rate of "moderate").

CHAPTER III

MODIFIED ASSUMPTION BASED TRUTH MAINTENANCE

A. Truth Maintenance Systems

When a transient occurs, it can be identified by an "expert" from the pattern made by the component states of nuclear power systems. In other words, in a given nuclear power plant similar general transient trends will be evident every time a particular transient occurs; thus, the problem of transient analysis is one of pattern recognition. As presented in Chapter II, rules describing these transient patterns can be derived so that an inference engine is able to perform diagnostic reasoning.

Truth maintenance systems[19] (TMS) extend traditional rule-based reasoning strategies to perform more efficient search and reasoning with inconsistent or incomplete information. The TMS manages truth through a nodal network of contexts representing a conclusion about the system. Dependencies of these contexts provide pathways through the search space. An inference engine interacts with a TMS to evaluate assumptions or rules that describe context nodes in the TMS search space. If a TMS node is "believed" then the TMS provides the inference engine with assumptions of contexts dependent on the original context.

A truth maintenance system associates a special data structure, called a node, with each problem-solver datum (which includes database entries, inference rules, and procedures). The basic nodal data structure of every truth maintenance system contains the datum with which it is associated (a descriptive node name), the justifications (all rules and procedures derivable for the given datum), and a label (a list including the datum and all antecedent datum). The datum is supplied by the problem solver (inference engine in most cases), and is never examined by the TMS. The justifications are supplied by the problem solver and are examined but never modified by the TMS. Justifications represent inference steps from combinations of nodes to another node (e.g., all rules and procedures derivable for the given datum). A node is believed if the justification is valid, in which case that node and all antecedent nodes contain noncontradictory assumptions (e.g., rules). The label identifies the node. The problem solver can add nodes, justifications and mark nodes as contradictory at any time.

The TMS provides two services: truth maintenance and dependency-directed backtracking. The TMS responds to the state of knowledge given in a knowledge base or qualitative model. This state of knowledge supports a set of premises which the TMS can process. Truth maintenance finds an assignment of belief statuses for every node such that every justification is satisfied. Although a node may have many justifications, only a few may hold, and one of these is chosen as the current supporting justification. A contradiction is encountered when a node, marked as contradictory, is assigned belief. Dependency-directed backtracking searches for the assumptions contributing to the contradiction. Contradictions found are removed from the TMS. A final solution derived from the TMS is found when no contradictions exist.

Reasoning using a TMS allows for a very flexible knowledge base. Concepts do not have to be specifically defined under a general context. Given a set of premises, the TMS will provide a best estimate solution. With increasing number of defined premises, the final solution becomes more specific.

While the TMS has the advantage of being able to find a solution for any given set of premises, other limitations make the TMS inadequate for system diagnosis. Multiple fault diagnosis and real time analysis are two goals of this expert system the TMS reasoning methodology cannot provide. The TMS is designed to converge to only one solution at a time (although modifying the state of knowledge can result in a new solution); however, many solutions may exist in a real system. The execution time of an expert system using TMS reasoning is directly related to the size of the TMS knowledge base; thus, a large TMS knowledge base requires longer computational time.

Although the TMS concept was not specifically designed for diagnosis purposes, it provides the foundation for an extension of the TMS that is capable of dealing with diagnosis, the assumption-based truth maintenance system.

B. Assumption Based Truth Maintenance System

The assumption based truth maintenance system[20] (ATMS), like the traditional TMS, is structured by a network of nodes describing a system. The major difference between the ATMS and a traditional TMS, is that ATMS context membership is a

subset of a general context; therefore dependency-directed backtracking is not necessary. The ATMS also allow multiple solutions because, unlike the basic TMS, nodes are contingent in a structured hierarchy.

Like a traditional TMS, processing knowledge about a system using an ATMS initially assumes that all possible solutions exist unless contradictions occur. In such a case that a node is declared contradictory, it is removed from the justification and knowledge processing continues until the justification is shown to be noncontradictory. Figure 1 displays the knowledge structure used by CATALisp (described in Chapter V). The node NOT.NORMAL represents all possible assumptions (i.e., all possible transient conditions) about the given system (i.e., the nuclear power plant). Figure 2 gives an example of the anatomy of a node used in this code. After the NOT.NORMAL node is processed, the nodes RCS failure, steam generator failure, SCS (secondary coolant system) failure, reactor trip, and LOSP (loss of offsite power) are examined. The environments for which these nodes hold are evaluated and if they are determined to be contradictory with the state of the system, they are removed from the justifications. For example, in the event that only the node RCS FAILURE is not contradictory, the justification is the nodal set {NOT. NORMAL, RCS FAILURE}. Figure 3 presents a portion of the ATMS nodal network, used in the software, in standard ATMS graphical notation (rectangles represent assumptions, ellipses represent premises or datum, and arrowheads represent justifications). This figure also displays the label for the Small LOCA node. Processing a node without contradictions results in the creation of an assumption or a set of assumptions. For example, when the RCS FAILURE node is processed, it creates the ATMS nodes LOCA and ATWS. This procedure is analogous to forward-chaining, however, it differs from forward chaining in its ability to reason with inconsistency and uncertainty. The assumptions (part of the environment processed by the inference engine) underlying these nodes are created as part of the nodes. For more complicated knowledge structures in which nodes might be cross referenced by superior nodes, the tasks are nontrivial and require more computation to process. The advantage of a more complicated knowledge structure is that instances of environments with the same consequence can be used. This is necessary because often one set of assumptions cannot adequately describe a consequence. For example, if a LOCA is occurring in the reactor coolant system, the pressure in the RCS usually decreases; however, because of the High Pressure Injection System (HPIS) safety system, the pressure can be rising when the HPIS is

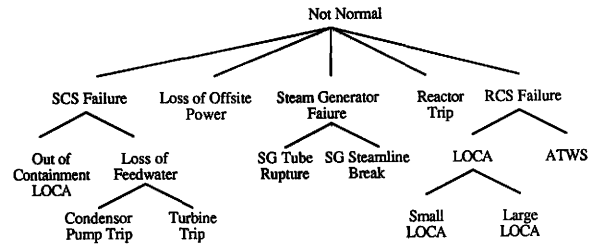


Figure 1. ATMS Knowledge Structure used by CATALisp

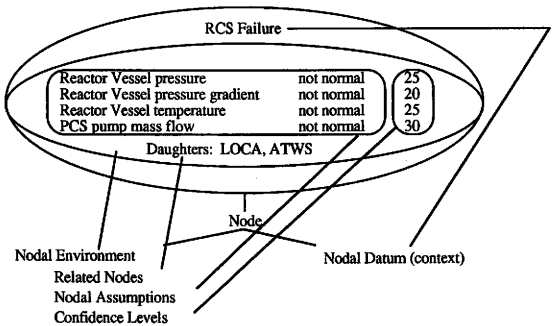
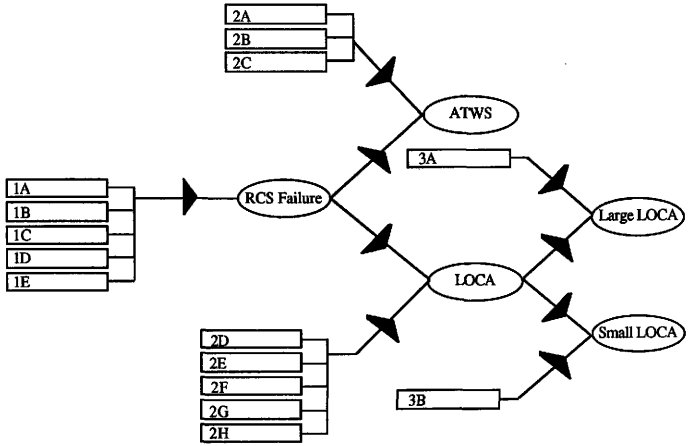


Figure 2. ATMS Nodal Anatomy



<Small LOCA,{3B,{2D,2E,2F,2G,2H,{1A,1B,1C,1D,1E}}}>

Figure 3. Sample ATMS Network in Standard ATMS Graphical Notation

on. Given these two instances have similar consequents, a knowledge structure might be created with both of these instances being represented as nodes. However, in complex systems such as nuclear power plants, many permutations of possible events can occur. A full knowledge base including all possible events would be very large and, hence, might require extensive computational attention. Therefore, one goal of this work was to simplify the knowledge structure without significant performance degradation, thus, extending this method to effectively handle real world problems such as real-time nuclear power plant diagnosis.

The efficiency of the ATMS is directly proportional to the number of environments it is forced to consider. Ultimately, the observed efficiency of the ATMS is a result of the fact that it is not that easy to create a problem which would force it to consider all possible solutions, rather it can only consider the assumptions which hold true.

The primary diagnostic module of CATALisp coordinates the activities of the inference engine which processes information from the ATMS knowledge base and the qualitative model (which receives simulation information directly). Figure 4 presents a graphical representation of this module. The inference engine receives the assumption sets from the ATMS knowledge base, matches the assumptions with knowledge from the qualitative model, information that is not available is communicated back to the ATMS knowledge base and the knowledge base is modified (physically, the assumptions that cannot be processed are removed from the knowledge base and confidence level values are renormalized, this is discussed in section III.F), and when an assumption set correctly describes the conditions presented in the qualitative model, the result is reported.

C. Real Time Diagnosis and Information Reduction

While some expert systems have addressed the issue of nuclear power plant transient diagnosis from the standpoint of "after-the-fact[4-6]," this work focuses on analyzing data as it would be presented during an actual unknown reactor transient. The difference between these two strategies is this: in "after-the-fact" diagnosis all diagnostic considerations are analyzed at one time after an event has occurred; and in real-time diagnosis, diagnostic considerations are analyzed upon repeatedly during an event. For true real-time analysis one would want to reduce the time between

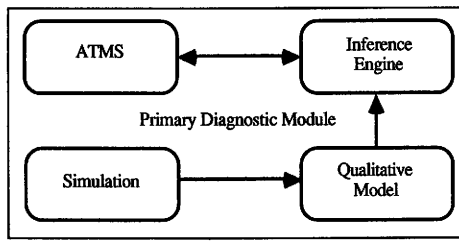


Figure 4. Primary Diagnostic Module

diagnostic iteration to be as small as possible; however, for transients that are not highly unstable, this iteration time step maybe larger. During highly unstable transients, a large time step might give meaningless results because many events could have occurred between iteration steps. Therefore, an iteration time of under 3 seconds would be considered good for most applications. Of the programs that have been reviewed that address diagnostics of real-time data, none have achieved the ideal of being able to obtain results instantaneously; rather, the authors have admitted that large improvement needs to be made in this area.

A major bottleneck in designing a real-time transient diagnosis and mitigation code is the abundance of information that is available for analysis. Much of this information is useful only for diagnosing specific transient cases; therefore, managing only that information that must be processed to obtain accurate results is a prime goal for efficient information handling.

In the nuclear power plant information is received in the control room from hundreds of locations in the plant. Information in the form of thermal-hydraulic data, annunciator status, and sensor data are available to an operator for consultation during operation. However, during transient scenarios only a small subset of this information is actually considered when the operator makes his final diagnosis. Therefore, to emulate an operator's actions during a transient, a study must first be made of procedures and thought processes an operator performs during these situations. Some studies have been performed in the this area.[1] One conclusion of these studies is that an operator reduces the information that must be examined in two ways. One way he does this is by performing sequential deductive reasoning for isolating a transient cause. This sequential deduction can be performed by using the ATMS.[20] As mentioned before, in an ATMS each datum or concept label is described with the sets of assumptions (rules or procedures representing the context of the datum) under which it holds. If the underlying assumptions hold true, the datum is assumed true. When this derivation is made, the ATMS records it in the most general way so that it covers as large a region of potential events as possible. For example, the RCS Failure node in Figure 1 represents a class of system failures found in the reactor coolant system. The idea is that the original assumptions are the primitive data from which all other data are derived. This method allows for a reduction in the number of assumptions that might have to be checked since only if a general context is verified will the ATMS check more specific problems. Such a database is said to be tree-structured since from general

context more specific contexts are derived. Different results are derived from different paths/branches taken in the database. By using this method, the objective of diagnosing both transient and failure location is accomplished systematically, similar to the method a human expert operator might follow to diagnose a transient.

The second way the operator reduces the amount of information he must process is to only consider information that define unique characteristics of the system's condition. For example, if the operator wants to know if the RCS is depressurizing, he only monitors one source, such as the pressurizer pressure indicator, rather than examining other instruments that can supply the same information, such as reactor vessel pressure or the pump outlet pressure. This procedure of monitoring on only one signal for indications of system condition allows for possible misdiagnosis if there has been an instrument failure. The method developed here presumes that all sensors have been previously validated. If sensor validation becomes a necessity, the assumptions defining the knowledge structure of the system can be expanded so that sensor validation can be performed to some degree. Redundant information useful for improving diagnosis should be included if it does not degrade performance significantly.

D. Diagnosis with Multiple Failure

Diagnostic reasoning requires the identification of deficiencies of a system on the basis of observed behavior discrepancies. In troubleshooting system failures, any differences in the presumed state of the system and the actual state of the system indicate that a problem could be present. However, the evidence does not always support a unique failure. Usually, additional signatures should be inferred to isolate the failure. It is important, for the sake of efficiency, to try to minimize the number of additional signatures that would have to be analyzed.

A significant problem that differentiates diagnosis of multiple failure from diagnosis of single failure is that simultaneously occurring failures can influence the system behavior patterns of events that would normally identify that individual event. Therefore, if one considers all permutations of possible events, the space of potential candidates grows exponentially with the number of faults involved. In this instance intense study must be performed to derive a set of assumptions that describe every possible event. This is not a reasonable consideration when dealing with complex

systems such as nuclear power plants. Therefore, an alternative method for dealing with multiple faults must be derived.

One possible method incorporates probabilities of failure in the diagnostic system analysis, so that results derive a set of most probable events. This methodology for getting multiple potential solutions has been demonstrated in the area of circuit diagnosis by applying probability of failure algorithm to an ATMS.[21] These probabilities are manipulated in such a way as to create a statistical distribution of possible causes of failure and their entropy, which is a value derived from the failure probabilities. Such methods assume that failure probabilities are known and are accurate. This is not the case in most real world problems.

Another problem that inhibits diagnosis is the possibility that information required for diagnosis is not available. However, it may be important that regardless on the amount of available information, a best-estimate evaluation be performed. To what degree is a best-estimate result valid is a very important issue in developing a best-estimate system. In considering this issue, advantages and disadvantages exist. The main advantage of such a system is that a solution is always available and independent of the amount of uncertainty present. However, in situations where much uncertainty exists, many solutions are generated, some of which may not actually be present.

Although the ATMS can derive multiple best-estimate solutions, the procedure cannot be applied efficiently to nuclear power plant diagnosis because it requires a complex ATMS knowledge base structure. However, to perform efficient, best-estimate reasoning, a modification to the ATMS knowledge processing has been made to simplify the ATMS knowledge base structure. This modification is based on "confidence levels", which are not to be confused with probabilities.

E. Confidence Level Assessment

A problem that occurs with simplifying the ATMS knowledge base is that a node can only be fired if the exact conditions described for that node are occurring. As mentioned before, for most real world cases exceptions can make up most of the knowledge base. Simplifying the knowledge base limits these possibilities greatly. However, if these nodes were capable of ignoring minor conflicts in the assumption sets, it would be possible to address these exceptions without actually describing them individually.

The focus of this modification is on how to process conflict resolution in the evaluation of assumption sets. Normally, an inference mechanism is used to evaluate the truth of the assumptions. If the set of assumptions are not valid, the context that they support is not valid. The basis of this modification is that under certain circumstances, minor conflicts should be ignored and the context should remain valid. Confidence level assessment allows this.

A confidence-level is the normalized confidence (0-1) that an "expert" expresses in a specific signature to describe a known transient. The confidence-level value for each transient is determined by examining the states of every relevant system in a nuclear reactor. The procedure for ignoring minor conflicts of assumption sets concerns how the assumptions are processed. Assigned to each assumption in the assumption set is a number that represents a weighing factor or confidence level. Confidence levels associated with each assumption in an assumption set are added together to arrive at an overall confidence-level for diagnosis of a particular transient. When a single event occurs, the sum of all the confidence levels is equal to 100%. However, should exceptions arise, any confidence level of 70% or greater validates the node. This cutoff value has been derived from an iterative process of evaluation of simulation results using different values. Accurately determining individual confidence levels is, therefore, the most important part in developing a successful transient analysis program by this method. The physical meaning of the confidence level is the percentage that the given assumption represents a particular transient relative to the other assumptions in the environment. In estimating the confidence level value, a thorough understanding of the transient is necessary. Given the list of conditions that apply to a particular transient, a valued judgement is made on the importance of individual conditional assumptions in the diagnosis of the transient. For the assumptions that apply to a diagnosis of the failure described by the context, the relative importance of each assumption is determined by extensively analyzing actual plant and code-calculated data (e.g., TRAC calculations). By comparing detailed transient simulations against heuristic results, estimates of the confidence-levels can be obtained. Interviewing experts in reactor safety could result in improved values for the confidence-levels and provide additional heuristics.[22]

Support for this method is as follows: Consider an event Q that can be described by the assumptions A, B, C, D, E, and F, an assumption (or fact) can be classified as either 1) absolute, one that holds for all circumstances of Q, or 2)

conditional, one that holds for most circumstances of Q; but may not hold under unusual conditions. For example, fact 7 in Table 1 is a conditional assumption because a LOCA could have resulted from high RCS Pressure; fact 9 is an absolute assumption because a LOCA event always contributes to containment radiation. When normalized weighing factors (confidence levels), a-f, are assigned to assumptions A-F, statements of fact can be derived about the truth of Q. If we consider that assumptions A-F represent absolute assumptions only, then the addition of all weighing factors with true assumptions, H, is equal to the maximum possible sum, H_{max}. If we select an arbitrary value, h, that represents the minimum value for which the summation of weighing factors of true assumptions provides a confirmation of event Q, we can extend this idea to include conditional rules. For the set of all absolute assumptions, H is always greater than or equal to h. In the set of assumptions that includes conditional assumptions, this rule does not hold for all possible cases. To confirm Q in this situation, all absolute assumptions should be true; however, the truth of all absolute assumptions may not alone verify Q. This assumptions implies that

$$H_{max} - \sum \text{Absolute Premises} < h \qquad \text{Eqn. 1.}$$

This assumption also implies that if an absolute assumption is not true and H is greater than h, Q is true. This implication is debatable, but this situation should not occur. A corollary to insure that this does not occur is that any absolute assumption weighing factor should be greater than all conditional assumption weighing factors. For the CATALisp program $H_{max} = 100\%$ and $h = 70\%$. An appropriate weighing factor or confidence level can be obtained from following these rules and by ranking the importance of the assumption against others in an assumption set.

To illustrate the procedures for deriving confidence levels for a LOCA transient, knowledge of the relevant system conditions that identify a LOCA must be known. These can be the following: RCS pressure, RCS pressure gradient, RCS pressure rate of change, RCS temperature, RCS temperature gradient, and containment vessel pressure and radiation. Absolute rules that relate how these quantities behave during a LOCA should be identified, then a corresponding confidence level value should be coupled with those rules. In this case RCS pressure gradient and containment radiation data are the most valuable to the identification of a LOCA because the occurrence of a LOCA immediately affects these two quantities. For the conditional rules confidence level values can be derived relative to each other once the "rank of usefulness" is

Table 1. LOCA Facts with Confidence Levels

RCS Failure

1) RCS Pressure is Not Normal	20
2) RCS Pressure Gradient is Not Normal	15
3) Level of the Pressurizer is Not Normal	10
4) RCS Temperature is Not Normal	20
5) RCS Temperature Gradient is Not Normal	15
6) RCP Mass Flow is Not Normal	20

LOCA

7) RCS Pressure is Low	10
8) RCS Pressure Gradient is Decreasing as long as the HPIS is off	20
9) Radiation in Containment is High	25
10) Pressure is Containment is High	15
11) Sump Pump Status is On	10
12) RCP Mass Flow is Low	10

determined. Factors that determine the usefulness of system data are: response time to a transient, availability, and uniqueness to a transient.

Table 1 displays facts with corresponding confidences levels, as used by CATALisp, that define a LOCA event. Rules 1-6 are fired to determine whether the failure source is in the RCS, while rules 7-12 are fired to determine whether the problem is a LOCA. When a LOCA event occurs, all of these rules should fire affirmatively, provided that no other abnormal event is occurring simultaneously. The individual confidence levels are added together for a total confidence level of 100% for both the RCS Failure and LOCA.

F. Confidence Level Assessment and Uncertainty

For circumstances in which information is unavailable, the assumptions that represent this information are removed from the assumption set and the confidence-levels are recalculated so that confidence levels are renormalized to 100%. This procedure renormalizes the confidence level value evenly among the remaining assumptions. Equation 2 defines this renormalization procedure. Where a_i^* is the new confidence level, a_i is the old confidence level and $\sum a_j$ is the sum of confidence level value of known assumptions.

$$a_i^* = a_i + \sum_{j=1}^{n} a_j * 100\% \qquad \text{Eqn. 2.}$$

In this way, unknowns are not completely ignored, they are just removed from the list describing a transient. For this reason and because assumptions can appear in many assumption sets, greater information on conditions of other parameters may be necessary to identify the transient. In certain cases all assumptions may be unknown. A best-estimate solution should provide such solutions as long as the nodes that precede that node are evaluated affirmatively. As mentioned before, in a highly uncertain environment many solutions can be possible. For this reason an "Uncertainty Screen" was created for diagnosis. This function is capable of screening out all results having uncertainty that is classified as "very high". The advantage of this function is that an operator can first examine diagnosis from known quantities in the plant; then, with the uncertainty screen off, he can get information of all possible events present,

provide the additional information to remove some uncertainty, and get an improved diagnosis.

Uncertainty is assigned to diagnostic results from the summation of the confidence levels of those assumptions in an assumption set that cannot be evaluated because of insufficient information. When this value is 0%, "very low" uncertainty is assigned; less than 30% and greater than 0% corresponds to "low" uncertainty; less than 70% and greater than 30% corresponds to "moderate" uncertainty; greater than 70% and less than 100% corresponds to "high" uncertainty; and 100% corresponds to "very high" uncertainty. Table 2 displays this uncertainty scale.

The uncertainty value does not modify the confidence level, since the confidence level is based strictly on what is known. Therefore, it is not unusual to have a diagnostic result with 100% confidence level and "high" uncertainty.

Table 2. Uncertainty Scale

Summation of Confidence Levels of Unknown Assumptions	Associated Uncertainty
0 %	Very Low
1 - 30 %	Low
31 - 69 %	Moderate
70 - 99 %	High
100 %	Very High

CHAPTER IV

MITIGATION STRATEGY

A. Current Methods

At commercial power plants operators are trained to follow emergency response operator guidelines specifically developed for the given plant. These guidelines provide transient mitigation procedures for a variety of transient situations. They are step by step instructions of actions and expected responses to problems including procedures to follow in case the expected result cannot be obtained. If the operator cannot trust that certain equipment will respond to his control, these guidelines are meaningless. To insure operator reliability in the system, the system is constantly examined for possible failure and future maintenance needs.

System design plays a major role in defining emergency response guidelines to mitigate a transient. Certainly, there are basic differences between plant types such as Pressurized Water Reactors, Boiling Water Reactors, and Liquid Metal Reactors; but, beyond the basic system design many other components are included to improve the safety factor of the system as a whole, while increasing the complexity. Many safety systems are incorporated into the final design to insure the integrity of the system. These designs are also built with redundant coolant flow pathways so that some failures can be isolated and bypassed. These added features, while contributing to the total safety of the system, contribute greatly to its complexity. Therefore, the function of the safety features represent an important consideration when developing transient mitigation strategies.

B. Operator Obstacles

Efficient response to a nuclear power plant transient is of utmost importance. Quick identification of a problem from observed symptoms can prevent further propagation of failure in the plant, possibly preventing a more severe problem. However, many factors, such as stress, invalid instrument readings, and information overload, can prevent the operator from performing efficiently.

Human factors considerations must be included into a system design to aid the operator in performing his job. The incident at Three Mile Island in 1979 demonstrated this need and has since prompted modification of the control room. Signals received in the control room are now presented to the operator in a more orderly fashion to prevent information overload and redundant signals. Another lesson learned from TMI is the importance of valid instrument readings. Instrument verification must be perform during transient conditions to insure that the operator receives the correct information so that he can provide the proper mitigation procedures that the condition calls for.

It is possible that an event can occur that is not specifically defined by operator guidelines or it is possible that a secondary failure can prevent the continuation of mitigation procedures. In these situations the operator must deduce similarities between the transient and the transients defined by the mitigation procedures. Incidents of multiple failure or severe accidents complicate this task and prevent the strict use of mitigation procedures because the guidelines are typically defined for non-severe single transient events. The operator must rely on his own experience and intuition. The priorities the operator has first is to protect the safety of the public. This can be interpreted as to prevent the release of significant amounts of radioactivity. The greatest source of radioactivity is in the reactor core, therefore, insuring the reactor core integrity is of greatest importance.

C. Automation of Mitigation Procedures

Automation of transient mitigation procedures can offer much improvement over the present emergency operator response process performed at nuclear power facilities. In light of the additional processing capability and speed by use of computers, such an innovation can create a working environment capable of drastically improving operator efficiency. The computerization of operator response guidelines create the opportunity for a direct link between the nuclear power plant control system and the expert system. In this way, inquiries made by the response guidelines about the system can be provided by the control system rather than by the operator, reducing operator error and time wasted on data gathering. This ability could also preclude operator selection of a particular recovery strategy that depended directly upon the availability or adequate performance of a downed or degraded piece of equipment or suggest alternative procedures to follow. Together with a program specifically for best-

estimate transient diagnosis, an improved diagnosis could be gained by first using the diagnosis program to determine probable failure and then following the operator response guidelines with additional diagnostic rules included; assurance of correct first estimate diagnosis could be confirmed or alternate procedures provided for the correct result. Other areas of potential improvements include the development of interactive plant simulations based on emergency response guidelines. This could provide better information on plant response to potential upset conditions and increased understanding of the impacts of operator actions on the response of the plant. Such a system would be very specific with respect to the actions required of the plant operator for any specified plant condition, it could be used to better assess the likelihood of operator error in recovery scenarios, while providing training for personnel.

Some progress has evolved from work performed in the area of computerization of mitigation procedures for nuclear reactor transients or related subjects.[9,23] However, the fact that knowledge of how to fix or correct something that needs attention is much more esoteric than actually understanding what is wrong with it presents a major problem. One result of this work is an investigation of the applicability of the Westinghouse Emergency Response Operator Guidelines (WEROG) as an expert system.[23] The WEROG resulted from a systematic development process supported by all major elements of the commercial nuclear industry and represents a comprehensive body of knowledge available to guide nuclear operation under transient and accident conditions. The basic format of these guidelines are IF/THEN/ELSE statements. A conclusion of this work was that although these guidelines were not intentionally developed for expert system implementation, they provide over 2500 rules that could easily be codified to some degree in an expert system. A simple extension of this concept has been implemented for this project.

Mitigation procedures for a set of "typically occurring events" have been transferred to a database. The structure of this database is very similar to the format of the WEROG, including instructions of actions and expected responses to problems and procedures to follow in case the expected result fails. Mitigation procedures are presented to a user as they appear in the WEROG, the user is prompted to confirm the condition presented, and the procedure continues, jumping to different procedure guidelines as necessary. If the guidelines request information that is available from the instruments and gauges used when diagnosing a transient, the code processes that procedure for the user.

CHAPTER V

SOFTWARE EXPERIMENT

A. Introduction

Software experiments are often performed to evaluate a physical system without actually having to build one. For example, transient analysis using thermal-hydraulic codes, such as TRAC, are used to evaluate safety and design of nuclear power plants. The purpose of this software experiment is to demonstrate the reasoning ability of CATALisp and evaluate the methods on which CATALisp is based. The intent of this project was the development of an aid to a reactor operator that could in an ideal sense, reside in a nuclear power plant control room and be hardwired to the instrument panel. However, for this experiment, a "real world" nuclear plant control panel, providing nuclear plant data, must be simulated by software.

CATALisp, a qualitative model, the ATMS knowledge base, and an interface between the simulated nuclear plant control panel and the qualitative model are requirements for this software experiment. Given the CATALisp software, the qualitative model and the ATMS knowledge base must be developed (CATALisp provides a mechanism for establishing a link between the simulated control panel and the qualitative model). Section V.E. and V.I. discuss in detail the ATMS knowledge base and the qualitative model, respectively, used in this software experiment.

B. Development History

The number of choices available to a code developer for hardware and software tools and programming methodologies are large. The selection of a complete set of development tools is dependent upon the requirements of the code being developed. Every tool has strengths and weaknesses. Code requirements must be weighed against the abilities of a tool; however, even the "best" set of tools may not be able to function well together or even be usable.

For these reasons development of the CATALisp software has been evolutionary. Due to limitations, availability problems, and incompatibilities for certain

hardware and software, changes in code design had to be made. Early attempts at developing an expert system for the purpose of diagnosing nuclear reactor transients began on an Apple Macintosh™ using the production rule language OPS5 integrated with LISP. This initial code version was capable of processing rules to recognize a loss of coolant accident (LOCA), steam generator tube rupture (SGTR), loss of feedwater (LOFW), and reactor trip transients, yet it required one minute to diagnose one transient time step. These results were quite inadequate due in part to the inflexibility of OPS5 and the incompatibility of the language compiler with the computer. As hardware availability allowed, the project was moved to a XEROX 1108™. On the XEROX execution time was improved to five seconds for every transient time step (a factor of 12). The expert system building tool KEE™ [24] (Knowledge Engineering Environment) was used as the inference engine under the control of an INTERLisp program. With the expansion of the number of transients evaluated and the number of components in the reactor system, iteration time increased to ten seconds. At that time, implementation of a diagnostic strategy for handling uncertainty was being developed. It was assumed that a component in an unknown state was in the worst possible condition. This assumption worked fine for some simulations, but was not strictly accurate. The combination of these two problems made it necessary to devise another method that would be faster and perform diagnosis better. After an evaluation of the code objectives and the results up to that time, it was determined that not only was a more powerful machine necessary; but also that the program had to be created from scratch, rather than with a expert system building tool. This final version has been developed on a Symbolics 3640 and is written entirely in Common Lisp.

C. Rule-Based Reasoning Strategies

Rule-based reasoning is by far the most widely used knowledge representation scheme. Rules take the form of IF/THEN or IF/THEN/ELSE statements and combine factual and heuristic domain knowledge. Individual IF/THEN rules are linked together to form rule chains, which can then be used to deduce information about a domain.

Inference engine mechanisms used with rule base structures can take many forms. Simple rule-based methods are search, forward-chaining, and backward-chaining. Truth maintenance systems are an example of a more advanced rule-based

reasoning strategy. Search methods are employed when rule-chains cannot be structured. In these cases rules are fired, often in a predefined sequence, in order to establish facts about a domain by querying the user. An inference engine evaluates to firing any and all rules that satisfy the given premises. Forward-chaining infers conclusions from facts in a database, typically starting with an initial set of information provided by the user. Using this method of inference, conditions of rules (the IF part) are examined, and if a rule's conditions are satisfied, the action (the THEN part) is performed. Backward-chaining attempts to find probable causes for a conclusion or goal sometimes supplied by the user, other times taken from a small catalog. If a rule conclusion is known to be true, the conditions that could have given rise to the rule conclusion become subgoals, and the inference engine tries to establish the validity of these subgoals recursively.

The rule-based expert system's inference engine has two parts: the rule interpreter and the scheduler. The role of the rule interpreter is to continually evaluate all rules in the system to locate those whose conditions are satisfied. The scheduler then orders the execution of all such rules.

Nuclear power plant diagnosis using rule-based reasoning has been performed to some degree.[4-6,11] REACTOR and ICS-EXPERT, as mentioned in the Literature Search, are two simple rule-based expert systems for this purpose. As mentioned in Development History for this project, a simple rule-based methodology was originally used. However, this methodology proved inferior for handling uncertainty and conflicts of fact. This occurs with simply rule-based methods because they assume that all information is available and accurate. A problem with this method is that if an event not specifically defined by the given rule set that exists, the inference engine will not be able to infer diagnosis. For this reason the more advance ATMS method was necessary for this project.

D. Codifying Expertise

Experts frequently perform their jobs intuitively, sometimes with little awareness of the logical thought sequence in their thinking and decision making processes. This apparent lack of strategy greatly inhibits solution explanations, justification, and the training of others. Computers do not comprehend such subtleties

as intuition and hunches, so the expertise must be delineated in terms of logically linked facts, rules, and heuristics to be used in an expert system.

In most circumstances an expert system developer is not an expert in the field he wants to describe with the expert system. For the novice capturing expert knowledge into heuristics requires extensive investigation into relevant books or manuals, analysis of test case studies, and interviews with experts. Strategies have been devised to identify knowledge that can be codified. The main objective of these strategies is to weigh the knowledge on the basis of its relevance to a particular task. Expert systems have been developed capable of analyzing data and generating rules from trends that are evident in the analysis.[25,26]

Extensive study has been directed for deriving general heuristics about nuclear power systems. Basic sources such as final safety analysis reports (FSAR) and emergency response operator guidelines for specific nuclear power plants contain a significant amount of this information. Test case results of nuclear power plant transients have been the greatest source of knowledge for this project. Extensive analysis has been performed on a wide range of transient scenarios and general trends have been identified to derive qualitative patterns of various failures that might occur in a nuclear power plant.

For complex systems, such as nuclear power plants, most of this knowledge codification deals with only a small number of the all the possible situations. This is because of exceptions that arise in the system. For example, during a LOCA transient occurring in the reactor coolant system, the pressure in the RCS will usually be decreasing; however, safety systems, such as the High Pressure Injection System (HPIS), have been put in place to prevent adverse conditions. At a certain check point (i.e., when the pressure drops to a predefined level) the HPIS turns on. One side effect of the HPIS being on is that the RCS pressure may no longer be decreasing, but rather increasing. This fact requires a more complicated heuristic rule to describe RCS pressure during a LOCA transient. This rule may resemble IF ((The PRESSURE of RCS is DECREASING) or (The STATUS of HPIS is ON)) THEN (Transient may be a LOCA). The presence of these exceptions represent the largest contribution of heuristics to an expert system.

Codifying expertise is not considered to be an exact science, therefore, it requires extensive iteration of eliciting knowledge from "expert" sources, documenting (or implementing) the knowledge, testing the knowledge by comparing the expert's

analysis against simulations, modifying the implemented knowledge as necessary and retesting. In developing rules for this expert system, this method was followed strictly to derive the present set of rules.

E. Codification Methods and Results

For this work many of the rules used for diagnosis were compiled from the results of other research endeavors;[4-6,11] however, some rules were derived from the South Texas Project final safety analysis report,[27] Westinghouse Emergency Operator Response Guidelines,[28] and analysis of plant conditions provided by best-estimate thermal-hydraulic codes such as TRAC.[29-33] Rules were verified by incorporating them into the CATALisp software, performing simulation, and assessing the accuracy of the results. If certain rules did not contribute to or detracted from the validity of the results, they were modified or removed. This iteration was repeated many times under different conditions to confirm that rules that worked for one simulation, also worked for other simulations.

The rules used in the CATALisp software were compiled into assumption sets describing both general and specific events. Table 3 presents these assumption sets with the understanding that the inference engine that evaluates the facts weighs each assumption with respect to its "rank of importance" relative to other assumptions in the set (Appendix B contains a detailed listing of this knowledge base, as used by CATALisp). This concept was explained in Chapter III. This knowledge base is a compilation of only abnormal transients. Normal transients such as power ramp and reduction are not classified here; however, because some assumption sets have similar conditions to this events, more specific assumptions describing these conditions are included.

Table 3. Codification Result

Fault: REACTOR COOLANT SYSTEM FAILURE
Description: The symptoms describe conditions which might exist if a failure has occurred in the RCS.
Assumptions:
1) RCS Pressure is Not Normal

Table 3 - continued

2) RCS Pressure Gradient is Not Normal
3) Level of the Pressurizer is Not Normal
4) RCS Temperature is Not Normal
5) RCS Temperature Gradient is Not Normal
6) RCP Mass Flow is Not Normal

Fault: REACTOR TRIP
Description: These symptoms describe the conditions which might exist during a reactor trip. Typically, only rule 6 should be necessary; however, additional rules are included to demonstrate aspects behind the theory for rule processing.
Assumptions:

1) RCS Pressure is Low
2) RCS Pressure Gradient is Decreasing
3) Control Rod Level is Very Low
4) Neutron Density is Decreasing
5) RCS Temperature Gradient is Decreasing
6) Reactor Trip Annunciator On

Fault: STEAM GENERATOR FAULT
Description: These symptoms identify conditions that might be present for problems with the steam generators.
Assumptions:

1) Steam Generator Level is Not Normal
2) Steam Generator Pressure is Not Normal
3) RCS Pressure is Low
4) RCP Mass Flow is Low

Fault: SECONDARY COOLANT SYSTEM FAILURE
Description: The symptoms describe conditions which might exist if a failure has occurred in the Secondary Coolant System (SCS).
Assumptions:

1) RCS Pressure is Not Normal
2) RCS Temperature is Not Normal

Table 3 - continued

3) SCS Pressure is Not Normal
4) Generator Power is Not Normal
5) Turbine Pressure is Not Normal
6) Feedwater Heater Temperature is Not Normal
7) Condenser Temperature is Not Normal

Fault: LOSS OF OFFSITE POWER
Description: Confirms the lack of available offsite power.
Assumptions:

1) Generator Power is Very Low
2) Turbine Tripped
3) Diesel Generator Power is Very Low

Fault: LOSS OF COOLANT ACCIDENT (LOCA)
Description: Symptoms describe plant conditions during a coolant line break in the RCS.
Assumptions:

1) RCS Pressure is Low
2) RCS Pressure Gradient is Decreasing as long as the HPIS is off
3) Radiation in Containment is High
4) Pressure in Containment is High
5) Sump Pump Status is On
6) RCP Mass Flow is Low

Fault: ANTICIPATED TRANSIENT WITHOUT SCRAM (ATWS)
Description: Symptoms describe plant conditions during an ATWS.
Assumptions:

1) No Reactor Trip
2) Pressurizer Level is High
3) RCS Pressure is High
4) Generator Power is High

Table 3 - continued

Fault: STEAM GENERATOR TUBE RUPTURE

Description: These symptoms describe typical plant conditions after a tube rupture in a steam generator.

Assumptions:

1) RCS Pressure is Low
2) Pressurizer Level is Low
3) Condenser Radiation is High
4) Containment Radiation is Normal
5) Feedwater Pump Mass Flow is Low
6) Steam Generator Level is High

Fault: STEAM GENERATOR STEAMLINE BREAK

Description: These symptoms describe typical plant conditions after a break has occurred in the steam line exiting the steam generator in the secondary coolant system.

Assumptions:

1) RCS Pressure is Low
2) Pressurizer Level is Low
3) Condenser Radiation is Normal
4) Containment Radiation is Normal
5) Feedwater Pump Mass Flow is High
6) Steam Generator Level is High

Fault: LOSS OF FEEDWATER

Description: These symptoms identify plant conditions when the feedwater supply has been lost.

Assumptions:

1) Mass Flow of the Feedwater Pump is Low
2) Steam Generator Level is Low
3) Emergency Feedwater Pump is On
4) RCS Temperature is High and Increasing unless Reactor has Tripped
5) Generator Power is Low

Table 3 - continued

Fault: OUT OF CONTAINMENT LOCA
Description: These symptoms define plant conditions following a break in the SCS.
Assumptions:

1) Turbine Pressure is Low
2) Mass Flow of Feedwater Pump is High
3) Steam Generator Level is Low

Fault: CONDENSATE PUMP TRIP
Description: Confirms condensate pump has tripped.
Assumption:

1) Condensate Pump Status is Off

Fault: TURBINE TRIP
Description: Confirms turbine trip
Assumptions:

1) Turbine Status is Off
2) Generator Power is Low
3) Turbine Pressure is Low

Fault: SMALL-BREAK LOCA
Description: Identifies size of RCS LOCA
Assumptions:

1) RCS Pressure is Decreasing "Slowly"

Fault: MEDIUM-BREAK LOCA
Description: Identifies size of RCS LOCA
Assumptions:

1) RCS Pressure is Decreasing "Moderately"

Fault: LARGE-BREAK LOCA
Description: Identifies size of RCS LOCA
Assumptions:

1) RCS Pressure is Decreasing "Quickly"

F. Real-Time Simulation

The goal associated with real time simulation of a nuclear power plant is to create an environment for the expert system that is as close to reality as possible. Assumptions made to this affect are the following: 1) not all information is readily available, due to component failures, some instruments may no longer function or may no longer give accurate results; 2) not all of this information needs to be received as time dependent data, certain relevant information that does not change over a period or is less critical can be included directly into the model; and 3) that a simulation can consist of either a large or small amount of both relevant and irrelevant information.

While real time simulation might imply an analog method of handling information, this is not possible for a standalone, single processor machine to perform. The desired environment for this situation is to simulate the process by which an expert system would handle information if it was directly connected to a plant control panel. Conceptually, this means that information would be received directly from the plant by a "signal control processor," which would store plant information directly into specific memory locations of the computer, so that this information could be directly retrieved from memory (see Figure 4). To simulate this process on the computer, simulation data is extracted in "snapshots." Information on plant conditions are called upon by the expert system as often as possible. Ideally, the time delay between iterations should be infinitesimal; however, for analyzing a nuclear power plant system this time delay can be as much as 3 seconds for most transients, with exception to highly unstable transients conditions, in which a 3 second delay may make a diagnosis meaningless. While the time delay factor is a function of how fast the inference engine can process the rules, knowledge of this fact is necessary for creating a transient simulation.

So that CATALisp can use this information, an assumption that the values from the first time step represent "normal" plant conditions is made. This assumption gives CATALisp flexibility for handling many different transient scenarios, because CATALisp can analyze data independent of the attribute value units. While this eliminates rules that contain quantitative constraints, this does not eliminate rules that contain quantitative constraints disguised as percentages of an arbitrary normal (i.e., If NEUTRON DENSITY is 5% of NORMAL, Then Reactor has been Tripped).

Sources for comprehensive sets of nuclear reactor transient simulation data are few. One candidate has been to use the code output from large thermal-hydraulic codes

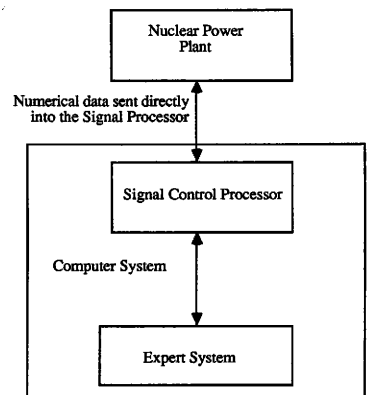


Figure 5. Ideal Expert System/Simulation Arrangement

such as TRAC, RELAP5/MOD2[34], etc. Another source of simulation data has come from reports of "test cases" using these codes. Many such studies have been performed at Los Alamos National Laboratory and Idaho National Engineering Laboratory using TRAC and RELAP5 and are readily available in journal paper form. Since this expert system is not directly coupled to a nuclear plant simulator computer, when using "test case" reports, the presented material must be first converted from the graphs and charts to data set as would be required by the expert system. By hand this is a strenuous task; however, as part of the expert system package developed, a code capable of allowing a user to reproduce these graphs and save transient simulation data as interpolated from given points on the graph was developed.

G. CATALisp Environment

CATALisp is a highly interactive and asynchronous program. The CATALisp package includes the main transient diagnosis and mitigation routines as well as a program for recreating transient data from graphs and a program for creating mouse-sensitive graphics of the modified ATMS knowledge base. Figure 5 presents the basic anatomy of the CATALisp program.

The initial stage of using CATALisp requires the user to establish the link between the qualitative model and the simulation data and to supply any other information required for the model. This is done from a mouse-sensitive window containing all of the components. Inciting any of the components creates a menu of attributes of that component. Selecting any of these items queries the user to supply the value (in symbolic form, i.e., high, low, increasing, etc.) for this attribute or the name of the simulation file containing this information. When all the available information is supplied to CATALisp, CATALisp is ready to begin analyzing it. Simply selecting the RUN THE SIMULATION option in the main menu begins this operation.

During the simulation run sequence, various information is available to the user. A gauge report supplies the relevant conditions of all components, that is, it displays all the information it has examined to make a diagnosis. Transient plots are available for those conditions which have been defined as transient data. A transient report is constantly updated to display the nodal labels of the modified ATMS that have been fired successfully. A more detailed transient report is available to explain what is happening. Diagnosis provides identification of specific problems in a given area and

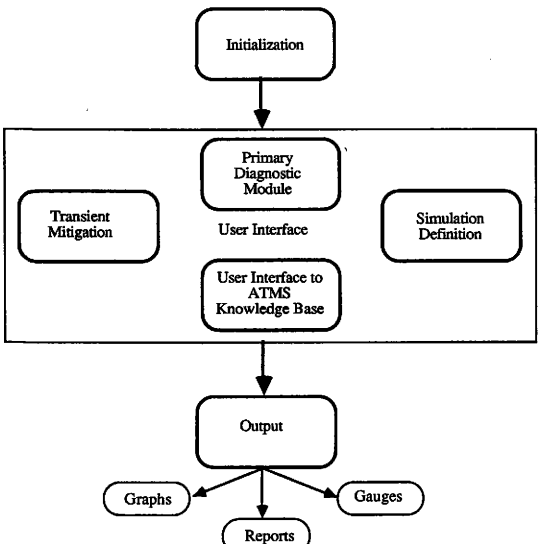


Figure 6. CATALisp Program Anatomy

the corresponding uncertainty of that event. The actual confidence level values associated with each node can also be retrieved from the options menu. During any simulation, additional information may be given to CATALisp by a process similar to that performed initially to start the simulation. When this is perform, a list of information that CATALisp believes to be relevant and not known is provided as a prompt for the user. The completion of his procedure requires that CATALisp evaluate a new set of confidence-levels for the newly learned information and repeats the analysis for an "improved" diagnosis. CATALisp has two modes of operation - normal and uncertainty screen. Uncertainty screen will not process nodes that contain assumptions that cannot produce a true or false result. Figure 7 display a typical screen dump of CATALisp during a simulation. Included in this figure are the gauge report, transient report, the diagnostic report, a transient plot, the lisp listener (provided for user responses to queries from CATALisp), and the options menu.

Once a final diagnosis has been derived, mitigation procedures can be called up as a separate process. In the operator response environment, the user selects, from a menu of transient events and operating procedures, the guidelines that address the transient previously diagnosed. Immediately, questions and instructions are addressed to the operator or, when appropriate, to the knowledge base on the state of the system and how to gain control of it. For multiple failures extra windows can be created to present different guidelines. Figure 8 presents a screen dump of the operator response program.

A useful tool available to the user at any time is the mouse-sensitive "tree" of ATMS nodes. Selecting any item in the tree creates a menu list of the assumptions/rules defined for that given node. The user can select any item in the list to see if the rule is true or false. Figure 9 provides an example of this feature. The menu shown in this figure displays assumptions under which a LOCA event is confirmed. Any of these assumptions can be selected for processing the truth of that assumption. The result is displayed in the top left corner of the window under the word "Examine". If the assumption is unknown, the displayed result is "Not.known", rather than "True" or "False".

The other feature available in CATALisp is the simulation development tool. Figure 10 shows a sample screen dump of this feature. A mouse-sensitive graph is created from constraints provided by the user, such as time length of transient and maximum property value. The graph can be easily recreated by depressing a mouse

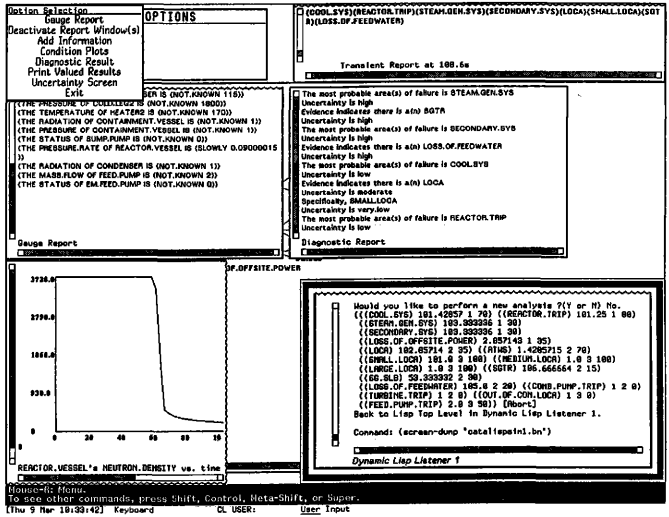


Figure 7. CATALisp Screen Dump

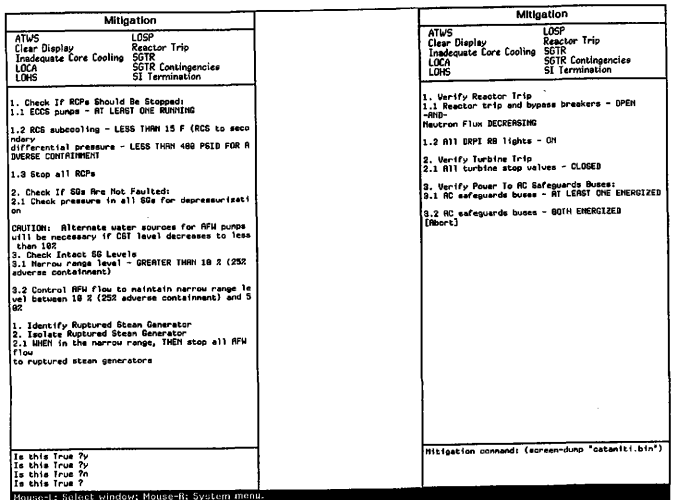


Figure 8. Screen Display of Emergency Response Mode

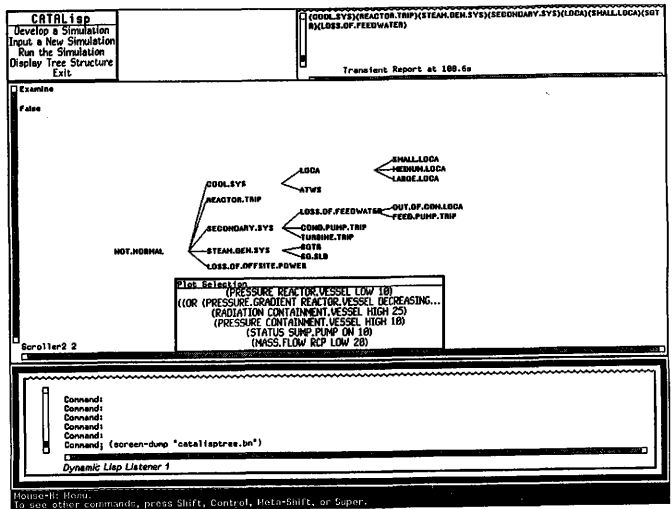


Figure 9. Screen Display of Mouse Sensitive ATMS Nodal Tree

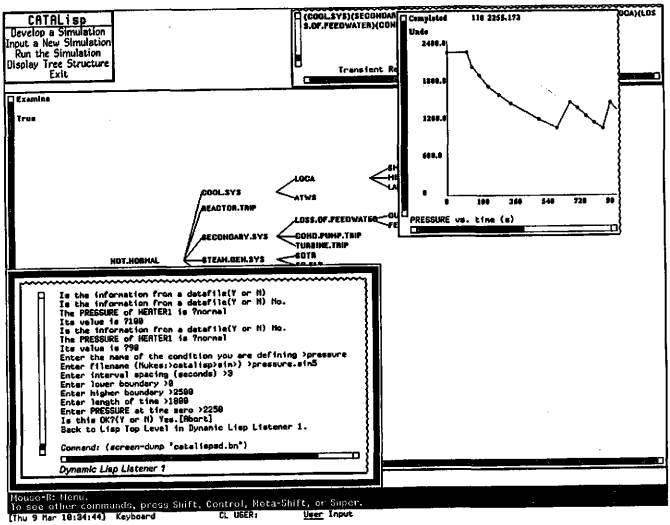


Figure 10. Screen Display of Simulation Developer Feature

button at a point on the graph. A second mouse button provides information on current coordinate location on the graph. At completion of the graph, a paired data set is created by linearly interpolating between points on the graph.

H. Physical Model

The original intent of this work was to specifically analyze Pressurized Water Reactors (PWR); thus, the qualitative model and heuristic rules are designed for this type of system. To test CATALisp a qualitative model (See Appendix C) of a simple one loop PWR was developed. It consists of 27 components and their relevant properties. Figure 10 displays this model. Of the 25 components, 14 components define "main" system components, 4 components define control or safety system components, 7 components define water and steam lines, and 2 components represent the containment vessel and the sump. Slots are attached to these components to store information that the control room might have on these components.

I. Case Studies

Four simulations have been developed and tested to demonstrate CATALisp's abilities. These simulations are driven by data sets defining the conditions on important nuclear plant components as a function of time. Global knowledge of a particular transient is put into the model to describe non-transient states in the system. Appendix A contains graphs of the transient data used in these four simulations.

The first simulation described a simple single-failure transient tripping a second at a certain time during the simulation. The data for this simulation was taken from output of a TRAC calculation for a primary side cold leg loss of coolant accident (LOCA) combined with a loss of feedwater (LOFW) at the South Texas Project Nuclear Plant.[35] Only RCS pressure, RCS temperature, HPIS mass flow, and core neutron density are supplied to CATALisp. Given this minimum set of information during a normal run, all possible results were displayed. While all results are viable, more information must be provided by the user to gain a more specific response of the events occurring. Specifically, information necessary to confirm or deny the presence of a SGTR is required, such as steam generator water level or condenser radiation. Table 4 presents the transient sequence and CATALisp results for this simulation.

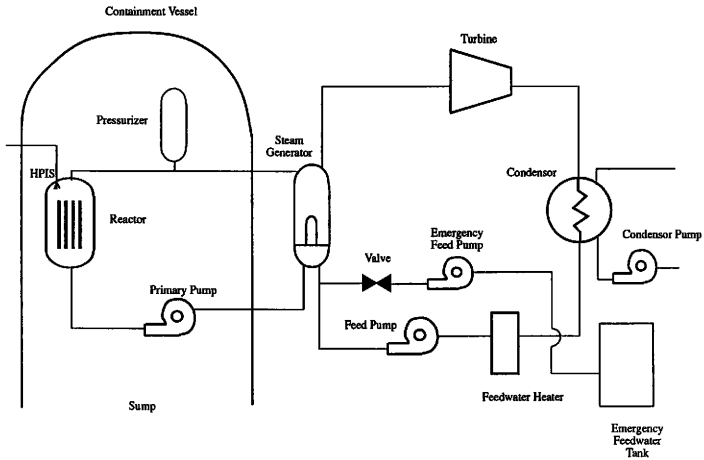


Figure 11. One Loop PWR Model Used for Testing CATALisp

Table 4. Transient Sequence and CATALisp Results for Simulation 1

Time	Event	CATALisp Result
0.00 s	main and auxiliary feedwater supplies cut	---
5.82 s	1 inch small break in primary side hot leg	---
10.6 s	---	LOFW (high, 100%)
15.9 s	---	Small LOCA (moderate, 100%)
18.6 s	---	SGTR (high, 100%)
62.0 s	reactor scram	Reactor Trip (low, 100%)
180 s	End of Transient	
	Uncertainty Screen Off	
0.0 s	---	Condensate Pump Trip
0.0 s	---	Out of containment LOCA
0.0 s	---	Turbine Trip
0.0 s	---	LOSP

The second simulation described a loss of offsite power combined with loss of feedwater transient. However, for this simulation only core average pressure and temperature data (reproduced from TRAC output[28]) were used. Under these conditions many results, depending on the state of the reactor, were possible. As in the first simulation, due to the large amount of uncertainty present, additional information is necessary for CATALisp to make the correct diagnosis, specifically that a LOFW has occurred. Extensive use of the ADD INFORMATION feature was performed to demonstrate how CATALisp can perform best-estimate analysis. Information from the reactor trip annunciator and steam generator level were later provided, proving that the ATWS was not present and providing greater certainty to the LOFW. Table 5 presents the transient sequence and CATALisp results for this simulation.

The third simulation was a recreation of conditions similar to that which occurred at the Three Mile Island Unit 2 (TMI-2) Reactor for the simple one loop reactor model. The TMI-2 facility is a Babcock and Wilcox, two loop reactor rated at 2772 MW thermal power. The sequence of events is extensive and spanned over many hours; but for this simulation, an abridged analysis was performed on only the first hour's events.[36] Because this model has only one loop, individual events occurring in the two loops of TMI-2 were compiled into one. Basically, the expected resulting transient was an initial loss of feedwater due to condensate pump trip, followed by a loss of coolant in the RCS due to the stuck open electromagnetic relief valve. The information used in this simulation was the following: RCS pressure, RCS temperature, neutron density, pressurizer water level, steam generator water level, condensate pump status, sump pump status and containment vessel radiation. Table 6 presents the transient sequence and CATALisp results for this simulation.

The last simulation examined the conditions from a single event, SGTR (from TRAC output[29]). RCS pressure, RCS temperature, steam generator water level, neutron density, containment radiation, condenser radiation, and pressurizer water level were provide for this simulation. Table 7 presents the transient sequence and CATALisp results for this simulation.

Table 5. Transient Sequence and CATALisp Results for Simulation 2

Time	Event	CATALisp Result
0.00 s	LOSP, LOFW	---
0.50 s	reactor scram	---
120 s	---	LOFW (high, 100%) ATWS (low, 100%)
494 s	PORV valve opens	
3000 s	End of Transient	
	Uncertainty Screen Off	
0.0 s	---	Condensate Pump Trip
0.0 s	---	Out of containment LOCA
0.0 s	---	Turbine Trip
0.0 s	---	LOSP

Table 6. Transient Sequence and CATALisp Results for Simulation 3

Time	Event	CATALisp Result
0.0 s	Condensate pump trip main feed pump and turbine trip	Condensate Pump Trip (very low, 100%)
6.0 s	Pressurizer relief value opens	---
8.0 s	Reactor scram	---
10.0 s	---	Reactor Trip (low, 100%)
13.0 s	Pressurizer relief valve does not respond to closure	---
15.0 s	Pressurizer water level peaks	---
38-40 s	Auxiliary feed valves fail to respond to 30 inch SG water level setpoint, due closed block valves	---
1 - 4 min.	Voids appear in core	---
80.0 s	---	Small LOCA (moderate, 80%)

Table 6 (continued)

2:02 min.	HPI low pressure setpoint reached	---
7:29 min.	Sump pump turns on	Small LOCA (moderate, 100%)
8:00 min.	Emergency feedwater restored	---
1 hour	End of Transient	---
	Uncertainty Screen Off	
0.0 s	---	Turbine Trip
0.0 s	---	LOSP

Table 7. Transient Sequence and CATALisp Results for Simulation 4

Time	Event	CATALisp Result
0.00 s	SGTG	---
15.0 s	---	SGTR (low, 100%)
838 s	reactor, turbine, and condensor trip	Reactor Trip (moderate, 82%)
858 s	Main feedwater trips	---
864 s	HPI begins	---
1 hour	End of Transient	
	Uncertainty Screen Off	
0.0 s	---	Condensate Pump Trip
0.0 s	---	Turbine Trip
0.0 s	---	LOSP

CHAPTER VI

CONCLUSIONS

A. Theoretical Methods Evaluation

The strength of the methodology developed here is the integration of the best estimate capabilities, especially in reference to multiple failure. For the degree of uncertainty presented from the simulations, this method for diagnosis has been demonstrated to be able to generate intelligent results based on known quantities. One possible disadvantage is that with high uncertainty, many results are possible; however, a reactor operator may have similar difficulties. From the case studies, the expected results are presented to the best degree of certainty. This ability is especially important when an operator is confronted with an event that is not specifically defined in emergency response guidelines (i.e., multiple failure, severe accidents, etc.). Provided with a best estimate evaluation of the situation, the operator can respond to the conditions presented in the evaluation or at least rule out certain event possibilities. Lessons learned from accidents, such as the TMI-2 incident, support the importance of being able to predict all possible events with the greatest degree of certainty. Future work in this area should stress this ability.

Confidence level assessment, as a modification to assumption based truth maintenance, has demonstrated an ability to ignore minor conflicts in assumption sets to validate contexts. For example, any result with a confidence level of less than 100% would not be validated in a tradiational assumption based truth maintenance system. If this was the case, simulation 3 and 4 results would be less accurate. The most dramatic result of confidence level assessment is that the ATMS knowledge base can be simplified, since exceptions to assumption sets describing a context do not have to be specifically defined. All rules describing a context can be contained under one context. This reduces the amount of tracing the ATMS has to perform, thus greatly improving performance efficiency and making real-time diagnosis possible. However, it is important that accurate confidence level values be assigned to individual premises in an assumption set. The rules governing this procedure (discussed in Chapter III) worked

well to a degree; however, iteration between creating rules and confidence levels, performing diagnosis through many simulations, and evaluating results was necessary to "fine tune" these values. A strict method for deriving these values has not been perfected; however, this study shows that even "close" values for the confidence-levels generate good results. Therefore, adding additional transients to the knowledge base requires that this iterative procedure be followed.

B. Software Experiment Results Evaluation

From the results of the four test cases performed using CATALisp, an assessment of CATALisp's performance can be made. The issues of focus in the evaluation are multiple failure identification, accuracy of results, uncertainty, and performance (including real-time and best-estimate diagnosis issues).

The LOFW combined with LOCA (simulation 1) and the LOSP combined with LOFW (simulation 2) simulations demonstrated CATALisp's ability to recognize multiple overlapping transients from a minimum set of information. CATALisp correctly diagnosed that the events expected where indeed present. It also showed that the information sets used in both simulations were inadequate to completely evaluate the transients. However, the fact that nearly accurate results were found suggest that the information that was supplied was very useful to the diagnosis. Additionally, simulation 2 showed how (when not enough information is provided) an incorrect result can have a greater certainty than the actual event. This leads to the conclusion that there is a limit were a minimum amount of information is absolutely necessary to make an accurate assessment of the situation. This has been shown to be the case with human operators.[37] For these two simulations uncertainty was high for all results, this was expected considering the amount of information provided.

The TMI transient (simulation 3) and the SGTR (simulation 4) simulations perform efficiently and accurately when provided with much more information. Given the extensive data supplied to CATALisp, CATALisp gives very good results under both uncertainty screen modes, considering that the transient signatures are defined in the ATMS knowledge base. With more information provided to CATALisp the possibility of conflicts in the assumption set increases due to certain uniqueness among similar transients. For this reason some results had lower confidence level values, however, still high enough to validate its context. Simulation 4 was included to demonstrate that for a single event with a sufficient amount of information supplied,

only that event is the result as expected (the conditions of a SGTR are often similar to a small LOCA).

General observation are evident in assessment of these simulations. With greater uncertainty and unknown information, confidence level values tend to be skewed to the extremes. These would be expected considering that only a subset of the total number of rules in an assumption set are fired. With less uncertainty there is a more even distribution of confidence level values. Also, with greater uncertainty accurate results were not confirmed immediately following the occurrence of a fault. For example, in simulation 1 the LOCA event was not reported until 10 s after it had occurred. This was because the rule "RCS PRESSURE is LOW" was false until 15.6 s. The uncertainty screen should probably be on at all times, except when the operator is confused about the conditions. With the uncertainty screen off CATALisp can provide a list of information that is necessary to perform a better analysis (from the ADD INFORMATION option). The operator could then supply this information for an improved diagnosis. For simulations 3 and 4 CATALisp was made to handle more information, yet, no significant difference was notice in CATALisp's performance. Under "real plant" conditions much information would be available to the computer and this simulation demonstrates the performance of this situation. One diagnositic iteration took about 1 s to perform.

Finally, a "necessary set" of information has been derived from these simulations that CATALisp should have during a simulation to provide accurate results with low uncertainty for the simple one-loop model. This set would likely be larger for a more detailed and complex model; however, this set would be the "backbone" to any "necessary set" of information. These are RCS pressure, RCS temperature, neutron density, reactor trip annuicator status, pressurizer water level, containment radiation, condenser radiation, steam generator level, turbine trip annuicator status, and HPI status. As this necessary set of information is processed, additional can be supplied to CATALisp when using the ADD INFORMATION option.

C. General Conclusions

Using knowledge-based systems in the nuclear power industry could offer improved safety, operation, maintenance, and efficiency of a nuclear power plant. Real-time analysis along with the integration of symptom-oriented diagnostic strategies

and mitigating procedures provides a powerful combination for analyzing and mitigating transients in nuclear power systems. Best estimate diagnostic methods provide a means for handling uncertainty and multiple failure occurrences.

The goal of any potential operator aid should address the issues of correct diagnosis, correct selection of required actions, or the transmittal of the necessary information to the operator. The results of this project indirectly present some evidence that "intelligent" computer aids can provide this invaluable service to an operator during transient scenarios. Extending this prototype and the qualitative model to analyze an actual plant could provide more valuable information. This requires modification of the ATMS knowledge base and the qualitative model only (see Appendix A and B). In this scenario CATALisp's response could be directly compared to an operator's response.

CATALisp is a software experiment that addresses most major issues involved in computer operator aids. These are: real time diagnosis, multiple failure transient, best estimate analysis, uncertainty management, information overload problems, and presentation of mitigation procedures. The results have demonstrated that symptoms in a plant can be accurately associated with transient events during complex transient conditions similar to a human operator's own reasoning in a reasonable time period. Even with a significant amount of uncertainty present, this program has shown a best estimate ability capable of resolving minor symptom conflicts of reasoning. It is not the intention of this expert system to do the job of the operator, this code still requires that the operator to provide it with the necessary information to assure an accurate analysis, to confirm its reasoning and to make a valued judgement from the diagnosis of the correct response.

D. Potential Areas of Application and Enhancements

The results from this code also imply new means for addressing transient analysis, personnel training, and severe accident situations. This tool could also be applied to similar systems such as that in the chemical industry.

D.1. Transient Analysis

Transient analysis of a nuclear power plant embodies study of predicting the behavior of a system from initiating events over time. This study is not limited to just

thermal hydraulic properties; but includes any condition that might be relevant to assuring the integrity of the plant (i.e., radioactive releases, availability of equipment, etc.). Extensive computations can be performed using system codes such as RETRAN, RELAP, and TRAC. The goal of transient analysis is to insure the integrity of a nuclear power plant by giving attention to prevention, mitigation, and response preparation of accident situations. Information provided in the study of transient behavior can identify potential breaches in safety so that design modifications or mitigation procedure can be made.

While traditional transient analysis deals with the prediction of state, intelligent computer-aided diagnosis presents another branch of transient analysis for prediction of transient cause. Results from computational transient analysis are used in the assessment of design and safety margins. Likewise, an unknown transient scenario could be run on a code similar to CATALisp to identify possible causes of undesirable phenomena; possibly leading to discoveries of inadequacies of design or equipment.

D.2. Training

The codification of expert knowledge of diagnosis and mitigation schemes in terms of logically linked facts, rules, and heuristics greatly enhances understanding of solution explanations. Expertise captured in this ordered form is vital for educating others of strategies and methods used by experts. An automated diagnosis system could serve in operator training courses demonstrating diagnostic reasoning, mitigation strategies, plant-operator interactions, and the qualitative behavior of nuclear systems under transient conditions. This system could be interfaced with an interactive plant simulator and respond to simulation input for an evaluation of the information. An operator trainee could then use this system to support or re-assess his own evaluation. An assessment of the likelihood of operator error during transient scenarios could also be made. During operator response and mitigation, this code could also provide qualitative information on plant response to potential abnormal conditions and increased understanding of the impacts of operator actions on the response of the plant. Considering that such a system would be interfaced with a simulator and based on emergency response guidelines, it could gather certain knowledge directly from the plant to determine whether the correct procedures were being followed by the operator; thus, it could be used to better assess the likelihood of operator error in recovery scenarios, while providing training for personnel.

D.3. Severe Accident Insights

Severe accident analysis and prevention is currently an issue of great concern in the nuclear industry. While nuclear power plants are designed with the safety principle of "Defense in Depth", these systems have a non-zero probability of failure. Severe accidents represent a class of low frequency events having high consequence to the plant and environment. Severe accidents are classified as accidents that present risk to containment integrity, thus, presenting a risk to the environment from core degradation or melting from inadequate core cooling. Inadequate core cooling can be due to a break in the reactor coolant system causing loss of water and/or steam, or a gradual escape of steam from the reactor coolant system causing heat to generate in the reactor core if the decay heat removal system is interrupted.

Many barriers already exist to prevent a severe accident situation. "Defense in Depth" barrier implies system redundancy exist, so that for certain failures, problems can be bypassed or isolated. Mitigation safety systems represent another design barrier for assurance of adequate cooling for the reactor. Emergency operator response procedures provide symptom-based diagnosis and response for cooling down a reactor. The containment presents a physical barrier for release of radioactivity resulting from a severe accident. Finally, response preparedness provides for attention to public concerns and cleanup. If the precursors of such events that cause inadequate core cooling are identified early in an initiating event, an accident capable of breaching these barrier might possibly be avoided. Studies have been performed to do just that by examining licensee event reports supplied to the Nuclear Regulatory Commission (NRC) by commercial power plants.[38] Precursors were identified on the basis of 1) If the event involved the failure of at least one system required to mitigate a loss of main feedwater, loss of offsite power, small-break LOCA, or steam-line break; 2) If the event involved the degradation of more than one system required to mitigate one of the above initiating events; or 3) If the event involved an actual initiating event that required safety system response. Codifying the symptoms to recognize these precursors is conceivable using CATALisp, thus allowing CATALisp the capability to serve as early warning for potential severe accidents.

D.4. Potential Enhancements

The current state of CATALisp is dependent on the qualitative model and heuristics describing the simple one loop PWR. New models can be created describing different plants or systems. Ideally, a general qualitative and heuristics model might be created so that new models would not have to be created from scratch. Alternatively, a preprocessor could be designed to ease the creation of new models. Diagnosis could possibly be improved by using component connectivity information in a model to define a physical location of a failure. Mass and energy balance equations could also be used to gain additional information about a plant. Communication between the machine and the user could be improved to provide more information, such as explanations of diagnosis, providing the importance of unknown information to diagnosis, and other useful plant information. Greater understanding of uncertainty could be gained by applying failure probabilities to events from Probablistic Risk Assessments. Mitigation procedures could be improved to provide intelligent conflict resolution when the establish operator guidelines do not adequately address a transient event.

REFERENCES

1. D. D. Woods, E. M. Roth, and H. Pople, *Modelling Human Intention Formation for Human Reliability Assessment*, Westinghouse Electric Corp., Pittsburgh, PA (1988).

2. Safety Code Development Group, *TRAC-pf1/mod, An Advanced Best-Estimate Computer Program for Pressurized Water Reactor Thermal-Hydraulic Analysis*, NUREG/CR-3858, LA-10157-MS, Los Alamos National Laboratory (1981).

3. R. G. Bowerman and D. E. Glover, *Putting Expert Systems into Practice*, Van Nostrand Reinhold Company Inc., New York (1988).

4. W. R. Nelson, "Reactor: An Expert System for Diagnosis and Treatment of Nuclear Reactor Accidents", *Proc. Nat. Conf. on Artificial Intelligence*, Pittsburgh, PA, 296, American Association on Artificial Intelligence (1982).

5. A. H. Wells and W. E. Underwood, "Knowledge Structures for a Nuclear Power Plant Consultant," *Trans. Am. Nuc. Soc.*, **41**, 41 (1982).

6. A. H. Wells, et. al., "Reactor Operator Diagnostic Ability Training Using Knowledge-Based Computer-Aided Instruction," *Trans. Am. Nuc. Soc.*, **46**, 42 (1984).

7. M. A. Bray, D. E. Sebo and B. W. Dixon, "Reactor Safety Assessment System--A Situation Assessment Aid for USNRC Emergency Response," *Expert Systems in Government Symposium*, Institute of Electrical and Electronics Engineers (1985).

8. J. A. Hassbergerl, "Simulation-Based Expert System for Nuclear-Power-Plant Diagnostics," University of Michigan, Ann Arbor, Dissertation (1986).

9. D. W. Miller, B. Chandrasekaran and D. Sharma, "Dynamic Procedure Synthesis, Execution, and Failure Recovery," *First International Conference on Applications for Artificial Intelligence to Engineering Problems*, South Hampton, Computational Mechanics Inc., Woborn, Massachusetts (1986).

10. D. W. Miller, et. al., "Intelligent Process Control Operator Aid -- An Artificial Intelligence Approach", *Proceedings of the Sixth Power Plant Dynamics, Control and Testing Symposium*, Knoxville, Tenn., University of Tennessee (1986).

11. B. Sun and R. Erdmann, "An Expert System Approach For Safety Diagnosis", *Nucl. Technol.*, **82**, 162-172 (1988).

12. J. E. Mott, "A Generalized System State Analyzer for Plant Surveillance", *Proc. Am. Nuc. Soc. Top. Mtg. on Artificial Intelligence and other Innovative Computer Applications in the Nuclear Industry*, Snowbird, Utah (1987).

13. A. M. Christie and R.W. Lindsay, "The DISYS Real-Time Diagnostics/Control System and its Application to the EBR-II", *Proc. Am. Nuc. Soc. Top. Mtg. on Artificial Intelligence and other Innovative Computer Applications in the Nuclear Industry*, Snowbird, Utah (1987).

14. B. W. Dixon, and M. F. Hinton, "Reviewing the Development of an Artificial Intelligence Based Risk Program," *Trans. Am. Nuc. Soc.*, **50**, 291 (1985).

15. J. E. Suich, "Logic Programming for Operational Analysis of the Savannah River Reactors," *Trans. Am. Nuc. Soc.*, **50**, 293 (1985).

16. P. J. Otaduy, "Demonstration of Expert Systems in Automated Monitoring," *Trans. Am. Nuc. Soc.*, **50**, 298 (1985).

17. D. E. Smith, L. F. Kocher and S. E. Seeman, "CLEO: A Knowledge-based Refueling Assistant at FFTF," *Trans. Am. Nuc. Soc.*, **50**, 292 (1985).

18. C. J. Puccia and R. Levins, *Qualitative Modelling of Complex Systems*, Harvard University Press, Cambridge, Massachusetts (1985).

19. J. Doyle, "A Truth Maintenance System," *Artificial Intelligence*, **12**, 231-272 (1979).

20. DeKleer, J., "An Assumption Based Truth Maintenance System," *Artificial Intelligence*, **28**, *2*, 127 (1986).

21. J. De Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, **32**, *1*, 97-130 (1987).

22. M. A. Meyer, and J. A. Booker, *Sources of Correlation Between Experts: Empirical Results from Two Extremes*, NUREG/CR-4814, LA-10918-MS, Los Alamos National Laboratory (1987).

23. M. E. Stella and H. V. Julian, "The Westinghouse Emergency Response Guidelines Considered as an Expert System," *Proc. Sec. Inter. Top. Mtg. on Nuclear Power Plant Thermal Hydraulics and Operations*, Tokyo, Japan, Atomic Energy Society of Japan (1986).

24. *Knowledge Engineering Environment*, ver. 2.1, Intellicorp, Mountain View, California (1985).

25. T. Washio, et. al., "Automated Derivation of Failure Symptoms for Diagnosis of a Nuclear Power Plant," *Annuals of Nuclear Energy*, **13**, *8*, 459-465. (1985).

26. J. Reifman and J. C. Lee, "Knowledge Base Generation for Power Plant Transient Diagnostics," *Proc. Third Inter. Top. Mtg. on Nuclear Power Plant Thermal Hydraulics and Operations*, Seoul, Korea (1988).

27. *Final Safety Analysis Report*, South Texas Project Unit 1 and 2, Vol. 10, Houston Power and Light Report, Houston, Texas (1985).

28. Westinghouse Owners Group, *Emergency Response Guideline Seminar*, Westinghouse Electric Corp., Pittsburgh (1981).

29. J. F. Dearing, et. al., *Dominant Accident Sequences in Oconee-1 Pressurized Water Reactor*, NUREG/CR-4140, LA-10351-MS, Los Alamos National Laboratory (1985).

30. B. Nassersharif, *Alternate Steam Generator Tube Rupture Mitigation Strategies for the Three Mile Island Unit 1 During a Loss-of-Offsite Power*, LA-UR-85-182, Los Alamos National Laboratory (1985).

31. B. Nassersharif, *Analysis of Multiple-Tube Ruptures in Both Steam Generators for the Three Mile Island-1 Pressurized Water Reactor*, LA-UR-85-1404, Los Alamos National Laboratory (1985).

32. N. S. DeMuth, et. al., *Loss-of-Feedwater Transients for the Zion-1 Pressurized Water Reactor*, NUREG/CR-2656, LA-9296-MS, Los Alamos National Laboratory (1982).

33. J. W.Bolstad, *Summary of TRAC Analyses of Steam Generator Tube Rupture Calculations for a Babcock and Wilcox Plant*, Interim Technical Report, Steam Generator Tube Rupture Overfill Study, FIN A7276 (1984).

34. V. H. Ransom,et.al., *RELAP5/MOD2 Code Manual*, NUREG/CR-4312, EGG-2396, EG&G Report (1985).

35. C. C. Guyot, "Analysis of Small-Break LOCA Transients Combined with Complete Loss-of-Feedwater Accidents for the South Texas Project Plant Using TRAC-PF1/MOD1," Master of Science Thesis, Texas A&M University (Dec 1987).

36. "Investigation into the March 28, 1979 Three Mile Island Accident by the Office of Inspection and Enforcement," NUREG-0600, U.S. Nuclear Regulatory Commission (1979).

37. J. W. Minarick, et. al., "Precursors to Potential Severe Core Damage Accidents: 1980 Status Report," NUREG/CR 4675, Vol. 5, U.S. Nuclear Regulatory Commission (1980).

38. A. J. Spurgin and R. L. Beveridge, "Lessons From Operator Responses to Accidents," *Proceedings from the ANS Topical Meeting on Computer Applications for Nuclear Power Plant Operation and Control*, Pasco, Washington, 305-309 (1985).

APPENDIX A

SIMULATION TRANSIENT DATA

Figure A-1 to Figure A-19 represent the transient data used in the four case studies presented in the Chapter V. It should be noted that not all the data used for these cases is presented in transient form, some data was also included representing "global" knowledge about the system during the transient. For example, for simulation 3 the reactor trip annunciator was on during the entire transient, therefore this knowledge was provided directly to the qualitative model, rather than in transient data form.

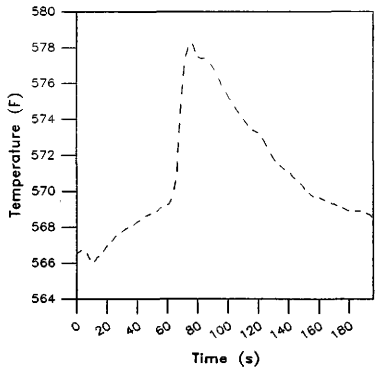


Figure A-1. Hot Leg Temperature vs. Time for Simulation 1

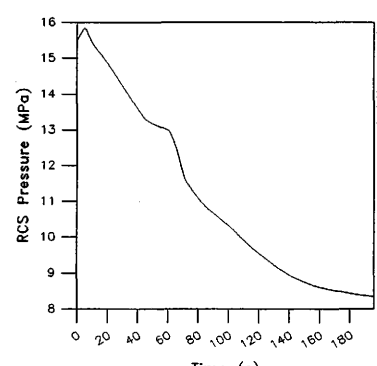


Figure A-2. RCS Pressure vs. Time for Simulation 1

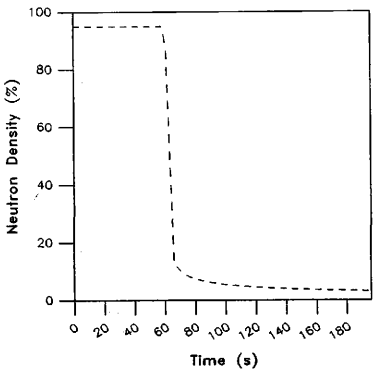


Figure A-3. Neutron Density vs. Time for Simulation 1

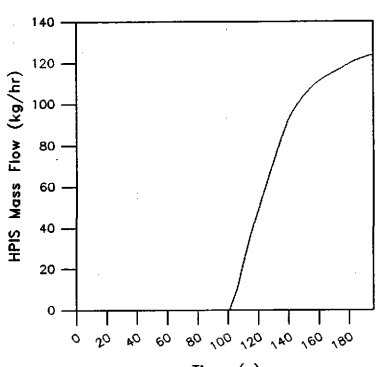


Figure A-4. HPIS Mass Flow vs. Time for Simulation 1

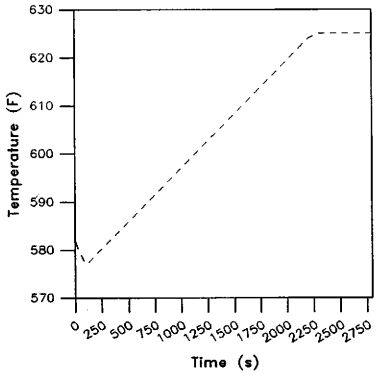


Figure A-5. Hot Leg Temperature vs. Time for Simulation 2

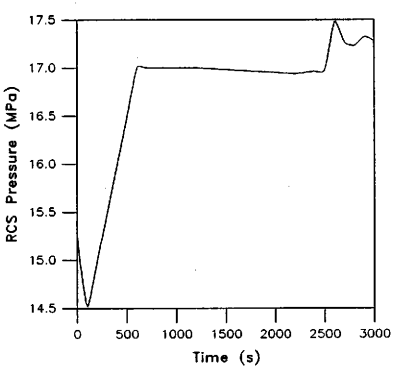


Figure A-6. RCS Pressure vs. Time for Simulation 2

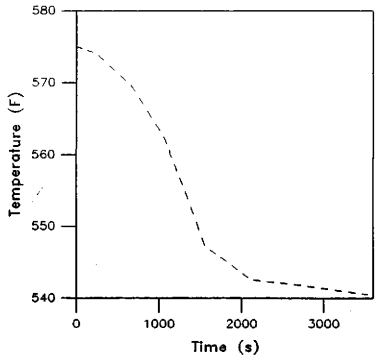


Figure A-7. Hot Leg Temperature vs. Time for Simulation 3

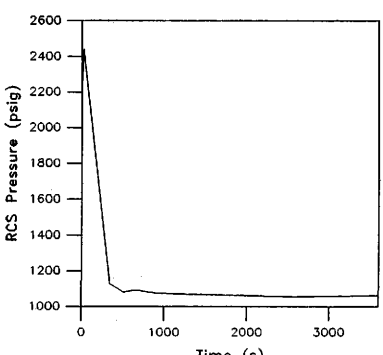


Figure A-8. RCS Pressure vs. Time for Simulation 3

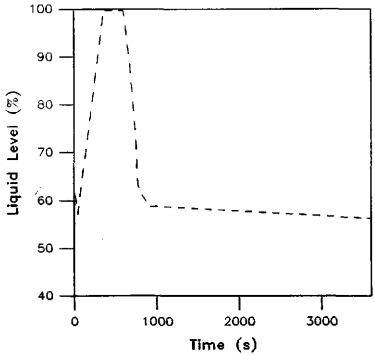


Figure A-9. Pressurizer Water Level vs. Time for Simulation 3

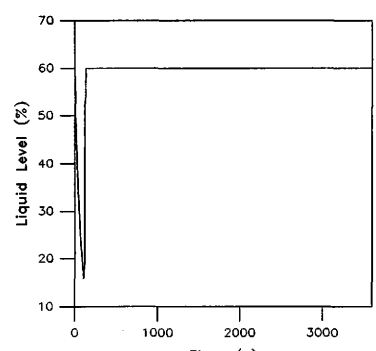


Figure A-10. Steam Generator Water Level vs. Time for Simulation 3

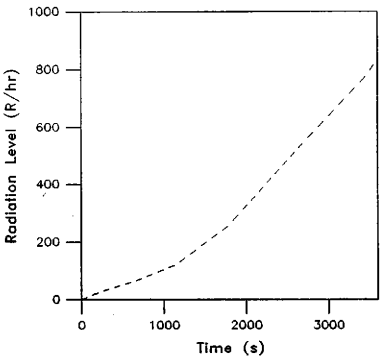


Figure A-11. Containment Radiation Level vs. Time for Simulation 3

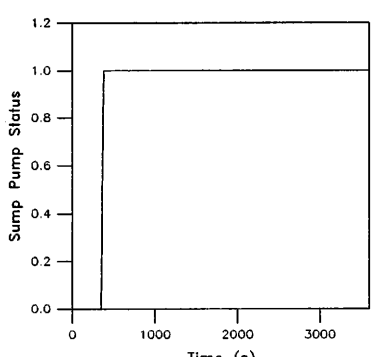


Figure A-12. Sump Pump Status vs. Time for Simulation 3

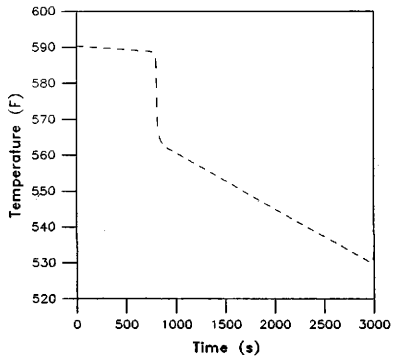


Figure A-13. Hot Leg Temperature vs. Time for Simulation 4

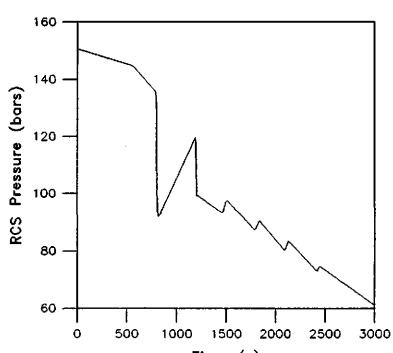


Figure A-14. RCS Pressure vs. Time for Simulation 4

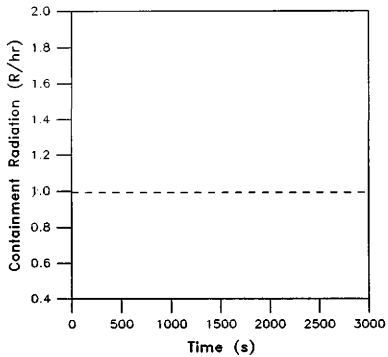


Figure A-15. Containment Radiation Level vs. Time for Simulation 4

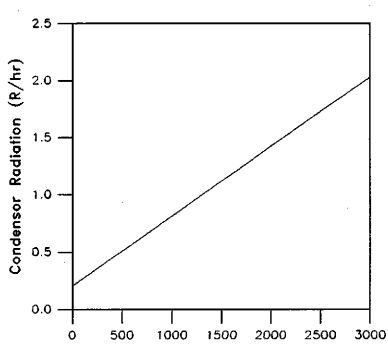


Figure A-16. Condensor Radiation Level vs. Time for Simulation 4

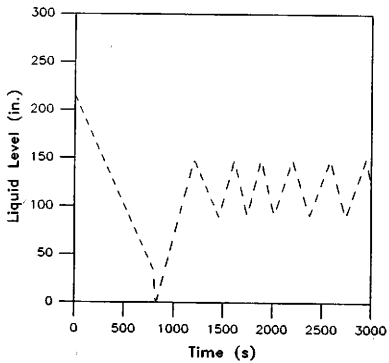


Figure A-17. Pressurizer Water Level vs. Time for Simulation 4

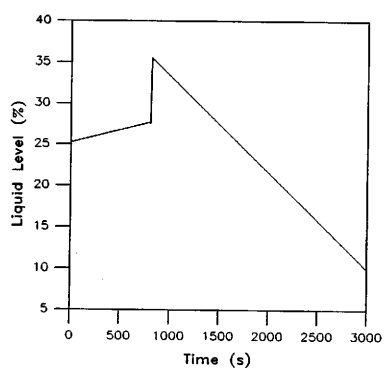


Figure A-18. Steam Generator Water Level vs. Time for Simulation 4

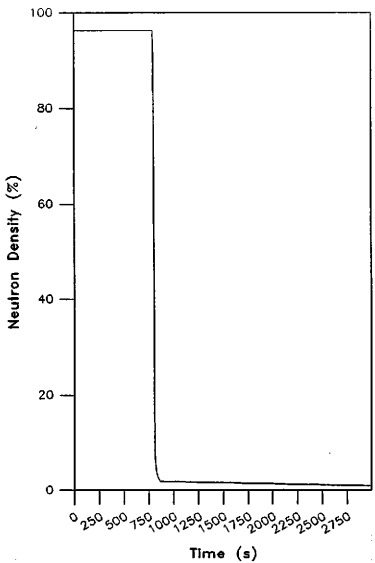


Figure A-19. Neutron Density vs. Time for Simulation 4

APPENDIX B

ATMS NODAL NETWORK

```
'((start)
  (not.normal
    ((pattern ((nil)))
    (daughter ((cool.sys) (reactor.trip) (secondary.sys)
               (steam.gen.sys) (loss.of.offsite.power)))))
  (cool.sys
    ((pattern ((pressure reactor.vessel not.normal 20)
               (pressure.gradient reactor.vessel not.normal 15)
               (level pressurizer not.normal 10)
               (temperature hot.leg1 not.normal 20)
               (temperature.gradient hot.leg1 not.normal 15)
               (mass.flow rcp not.normal 20)))
     (daughter ((loca)
                (atws)))))
  (reactor.trip
    ((pattern ((status reactor.trip.ann on 50)
               (pressure reactor.vessel low 10)
               (pressure.gradient reactor.vessel decreasing 8)
               (cr.level reactor.vessel low 10)
               (neutron.density reactor.vessel decreasing 15)
               (temperature.gradient hot.leg1 decreasing 7)))
     (daughter ((nil)))))
  (steam.gen.sys
    ((pattern ((level steam.gen1 not.normal 20)
               (pressure steam.gen1 not.normal 20)
               (mass.flow rcp not.normal 30)
               (pressure reactor.vessel low 30)))
     (daughter ((sgtr) (sg.slb)))))
  (secondary.sys
    ((pattern ((pressure reactor.vessel not.normal 15)
               (temperature hot.leg1 not.normal 15)
               (pressure turbine not.normal 15)
               (power generator not.normal 15)
               (temperature condenser not.normal 15)
               (pressure cold.leg2 not.normal 15)
               (temperature heater2 not.normal 10)))
     (daughter ((loss.of.feedwater) (cond.pump.trip)
(turbine.trip)))))
  (loss.of.offsite.power
    ((pattern ((power generator very.low 50)
               ((rule (probability turbine.trip > number 70.0)) 35)
               (power diesel very.low 15)))
     (daughter ((nil)))))
  (loca
    ((pattern ((pressure reactor.vessel low 10)
               ((or (pressure.gradient reactor.vessel decreasing)
                    (rule (mass.flow hpis > number 0.0))) 25)
               (radiation containment.vessel high 25)
```

```
                (pressure containment.vessel high 10)
                (status sump.pump on 10)
                (mass.flow rcp low 20)))
      (daughter ((small.loca) (medium.loca) (large.loca)))))
    (small.loca
      ((pattern ((pressure.rate reactor.vessel slowly 100)))
       (daughter ((nil)))))
    (medium.loca
      ((pattern ((pressure.rate reactor.vessel moderately 100)))
       (daughter ((nil)))))
    (large.loca
      ((pattern ((pressure.rate reactor.vessel quickly 100)))
       (daughter ((nil)))))
    (atws
      ((pattern (((rule (probability reactor.trip < number 70.0))
40)
                (level pressurizer high 15)
                (pressure reactor.vessel high 30)
                (power generator high 15)))
       (daughter ((nil)))))
    (sgtr
      ((pattern ((pressure reactor.vessel low 15)
                (level pressurizer low 15)
                (radiation condenser high 20)
                (radiation containment.vessel normal 15)
                (mass.flow feed.pump low 20)
                (level steam.gen1 high 15)))
       (daughter ((nil)))))
    (sg.slb
      ((pattern ((pressure reactor.vessel low 15)
                (level pressurizer low 15)
                (radiation condenser normal 20)
                (radiation containment normal 15)
                (mass.flow feed.pump high 20)
                (level steam.gen1 is high 15)))
       (daughter ((nil)))))
    (loss.of.feedwater
      ((pattern ((mass.flow feed.pump low 30)
                (level steam.gen1 low 10)
                (status em.feed.pump on 10)
                ((or (and (temperature hot.leg1 high)
                          (temperature.gradient hot.leg1 increasing))
                     (and (temperature hot.leg1 high)
                          (rule (probability reactor.trip >= number
70.0)))) 20)
                (power generator low 30)))
       (daughter ((out.of.con.loca) (feed.pump.trip)))))
    (out.of.con.loca
      ((pattern ((pressure turbine low 34)
                (mass.flow feed.pump low 33)
                (level steam.gen1 low 33)))
       (daughter ((nil)))))
    (feed.pump.trip
      ((pattern ((mass.flow feed.pump very.low 35)
                (status em.feed.pump on 15)
```

```
          (pressure feed.pump low 40)))
  (daughter ((nil)))))
(cond.pump.trip
  ((pattern ((status cond.pump off 100)))
  (daughter ((nil)))))
(turbine.trip
  ((pattern ((status turbine off 60)
          (power generator very.low 20)
          (pressure turbine low 20)))
  (daughter ((nil)))))
(end)))
```

APPENDIX C

PWR QUALITATIVE MODEL

```
'((start)
(reactor.vessel
  ((pressure (not.known 2250))
  (pressure.gradient (not.known 0))
  (pressure.rate (not.known 0))
  (neutron.density (not.known 80))
  (cr.level (not.known 80))
  (adj.sys (hot.leg1 cold.leg1))))
(hot.leg1
  ((pressure (not.known 2250))
  (pressure.gradient (not.known 0))
  (pressure.rate (not.known 0))
  (temperature.gradient (not.known 0))
  (temperature (not.known 650))
  (mass.flow (not.known 20))
  (adj.sys (reactor.vessel pressurizer steam.gen1))))
(pressurizer
  ((pressure (not.known 2250))
   (temperature (not.known 650))
   (level (not.known 50))
   (adj.sys (drain.tank hot.leg1))))
(hpis
  ((pressure (not.known 2250))
   (mass.flow (not.known 0))
   (adj.sys (hot.leg1))))
(steam.gen1
  ((level (not.known 85))
   (pressure (not.known 1185))
   (adj.sys (hot.leg1 cold.leg1.2 hot.leg2 cold.leg2
             emer.fw.sl1))))
(cold.leg1.2
  ((pressure (not.known 3))
   (pressure.gradient (not.known 0))
   (pressure.rate (not.known 0))
   (temperature.gradient (not.known 0))
   (temperature (not.known 550))
   (mass.flow (not.known 20))
   (adj.sys (cool.pump steam.gen1))))
(cold.leg1.1
  ((pressure (not.known 2000))
   (pressure.gradient (not.known 0))
   (pressure.rate (not.known 0))
   (temperature (not.known 550))
   (temperature.gradient (not.known 0))
   (mass.flow (not.known 20))
   (adj.sys (cool.pump reactor.vessel))))
(rcp
  ((pressure (not.known 2000))
```

```
   (mass.flow (not.known 250)))
  ((adj.sys (cold.leg1.1 cold.leg1.2))))
(emer.fw.sl1
  ((pressure (not.known 2000))
   (adj.sys (steam.gen1 emer.fw.pump))))
(emer.fw.pump
  ((adj.sys (emer.fw.sl1 emer.fw.sl2))))
(emer.fw.sl2
  ((pressure (not.known 1))
   (adj.sys (con.tank emer.fw.pump))))
(con.tank
  ((water.level (not.known 85))
   (water.level.gradient (not.known 0))
   (adj.sys (emer.fw.sl2))))
(hot.leg2
  ((pressure (not.known 1800))
   (pressure.gradient (not.known 0))
   (pressure.rate (not.known 0))
   (temperature (not.known 610))
   (temperature.gradient (not.known 0))
   (mass.flow (not.known 20))
   (adj.sys (steam.gen1 turbine))))
(turbine
  ((pressure (not.known 1800))
   (temperature (not.known 610))
   (status (not.known 1))
   (adj.sys (generator hot.leg2 condenser))))
(generator
  ((power (not.known 90))
   (adj.sys (turbine))))
(condenser
  ((pressure (not.known 1))
   (temperature (not.known 115))
   (radiation (not.known 1))
   (adj.sys (turbine cond.pump))))
(cond.pump
  ((status (not.known 1))
   (adj.sys (heater1 condenser))))
(heater1
  ((pressure (not.known 2))
   (temperature (not.known 150))
   (adj.sys (feed.pump cond.pump))))
(feed.pump
  ((mass.flow (not.known 2))
   ((adj.sys (heater1 heater2)))))
(em.feed.pump
  ((status (not.known 0))
   ((adj.sys (heater1)))))
(heater2
  ((pressure (not.known 1800))
   (temperature (not.known 170))
   (adj.sys (feed.pump cold.leg2))))
(cold.leg2
  ((pressure (not.known 1800))
   (pressure.gradient (not.known 0))
```

```
   (pressure.rate (not.known 0))
   (temperature (not.known 170)
   (temperature.gradient (not.known 0))
   (mass.flow (not.known 20))
   (adj.sys (steam.gen1 heater2)))))
(containment.vessel
  ((pressure (not.known 1))
   (radiation (not.known 1))
   (adj.sys (nil))))
(sump
  ((water.level (not.known 10))
   (adj.sys (nil))))
(sump.pump
  ((status (not.known 0))
   (adj.sys (nil))))
(diesel
  ((power (not.known 0))
   (adj.sys (nil))))
(reactor.trip.ann
  ((status (not.known 0))
   (adj.sys (nil))))
(end)))
```

APPENDIX D

CATALISP PROGRAM

```
COMMENT:  BEGIN is the initiation command for CATALISP

(defun begin ()
  (cond ((y-or-n-p "Would you like to perform a new analysis ?")
         (setq system nil)
         (setq cond1 nil)
         (setq time nil)
         (setq attr nil)
         (kb1)))
  (setq cn3 nil)
  (prim)
  (send *window* :deactivate)
  (send *window2* :deactivate)
  (send *window1* :deactivate)
  (send *window3* :deactivate)
  (send *result* :deactivate)
  (send *mouse* :deactivate)
  (send *standard-output* :select))

(defun prim ()
  (send large-win :expose)
  (send primary ':expose)
  (send primary ':choose)
  (send primary ':bury))


COMMENT:  VSNORM, GRADIENT, AND RATEET convert numerical data to
symbolic representation

(defun vsnorm (t2 nm)
  (prog nil
        (if (eql t2 nil) (return '(not.known)))
        (if (> t2 nm) (return '(high)))
        (if (> t2 2355) (return '(very.high)))
        (if (< t2 (* 0.05 nm)) (return '(very.low)))
        (if (< t2 nm) (return '(low)))
        (if (= t2 nm) (return '(normal)))))


(defun gradient (t1 t2)
  (prog nil
        (if (eql t2 nil) (return '(not.known)))
        (if (< t2 t1) (return '(decreasing)))
        (if (> t2 t1) (return '(increasing)))
        (if (= t2 t1) (return '(normal)))))

(defun ratedet (t1 t2)
  (let ((xxx nil))
    (prog nil
```

```
        (if (eql t2 nil) (return '(not.known)))
        (setq xxx (abs (- t2 t1)))
        (if (= xxx 0) (return '(zero)))
        (if (and (> xxx 0) (< xxx (* 5 cnn))) (return '(slowly)))
         (if (and (>= xxx (* 5 cnn)) (< xxx (* cnn 12))) (return
'(moderately)))
        (if (>= xxx (* 12 cnn)) (return '(quickly))))))

(setq kb (make-array 5))

COMMENT:  READER reads transient datafiles

(defun reader  (st co)
 (setq st (string-append "NUKES:>catalisp>" st))
 (with-open-file (stream st ':direction ':input)
   (let ((n 0))
   (prog nil
       label
          (setf (aref time n) (read stream nil))
          (setf (aref attr co n) (read stream nil))
          (if (equal (aref time n) nil) (return))
          (setq n (+ n 1))
          (go label)))))


COMMENT:  KBCALL allows I/O access to a knowledge base


;;;       If i = 1 kbcall inqueries the database
;;;       If i = 2 Kbcall writes to the database

(defun kbcall (r v i)
  (let ((a nil)    (b nil)      (c nil)      (b1 nil) (a1 nil) (at2 nil)
(at1 nil) (b3 nil)
      (entry nil) (fentry nil)  (bt nil)  (b2 nil)  (at nil)  (a2 nil)
(a3 nil)
      (kb1 nil) (b21 nil))
 (prog nil
  (cond ((eql (aref kb r) nil) (setf (aref kb r) '((start) (end)))))
  (setq a (list (cadr v)))
  (setq b (cadddr v))
  (setq c (list (car (reverse v))))
  (setq b1 (list '(start)))
  (setq a1 (list b))
  (setq at2 b1)
  (setq at1 nil)
  (setq b3 nil)
  (setq kb1 (aref kb r))
  (setq entry (append a c))
  (setq fentry (append (list b) (list (list entry))))
flag1
  (setq bt (caar kb1))
  (setq b3 (cdr kb1))
  (cond ((eql b bt) (setq at (cadar kb1)) (setq at1 at) (go flag2)))
  (cond ((and (eql (caar b3) 'end) (not (= i 1)))
```

```
      (setf (aref kb r) (append b1 (list fentry) b3))
      (return)))
  (cond ((and (eql (caar b3) 'end) (= i 1)) (return)))
  (setq kb1 (cdr kb1))
  (setq b1 (append b1 (list (car kb1))))
  (go flag1)
flag2
  (cond ((and (eql (car at2) nil) (not (= i 1)))
       (setq b2 (append (list (car a1)) (list (append at (list
entry)))))
      (setf (aref kb r) (append (reverse (cdr (reverse b1))) (list
b2) b3))
      (return)))
  (setq at2 (list (caar at1)))
  (setq a3 (cdr at1))
  (setq a1 (append a1 (list (car at1))))
  (cond ((equal a at2) (go flag3)))
  (setq at1 (cdr at1))
  (if (and (eql at1 nil) (= i 1)) (return))
  (go flag2)
flag3
  (if (= i 1) (setq c (cdar at1)))
  (cond ((= i 1) (return c)))
  (setq a1 (reverse (cdr (reverse a1))))
  (setq b1 (reverse (cdr (reverse b1))))
  (setq a2 (append a c))
  (setq b2 (append a1 (list a2) a3))
  (setq b21 (car b2))
  (setq b2 (append (list b21) (list (cdr b2))))
  (if (eql i 3) (setq b2 (append a1 (list a3))))
   (cond ((eql (cadr b2) nil) (setf (aref kb r) (append b1 b3))
(return)))
  (setf (aref kb r) (append b1 (list b2) b3)))))
```

COMMENT: UNKNOWN-RULE, ORGANIZE, AND FINDUNKNOWN derive a list of plant conditions that are unknown, yet important for diagnosis

```
(defun unknown-rule (ss1)
  (prog (a2 a1 dum0 dum1 dum2)
      (setq a2 ss1)
      (setq a1 nil)
     label
      (if (equal a2 nil) (return ss1))
      (setq dum0 (list (car a2)))
      (setq a2 (cdr a2))
      (cond ((equal (caar dum0) 'rule)
             (setq dum0 (car dum0))
               (setq dum1 (list (append (list (second dum0)) (list
(third dum0)) (list (first dum0)) (list (/ (seventh dum0) 2.0)))))
               (setq dum2 (list (append (list (fifth dum0)) (list
(sixth dum0)) (list (first dum0)) (list (/ (seventh dum0) 2.0)))))
             (setq ss1 (append a1 dum1 dum2 a2))
             (setq dum0 (append dum1 dum2))))
      (cond ((= (length (car dum0)) 2)
             (eval-rule (car dum0))
```

```
            (setq ss1 (append a1 col a2))
            (setq dum0 col)))
      (setq a1 (append a1 dum0))
      (go label)))


(defun organize (ss1)
  (let ((ssd1 nil)       (dup2 nil)   (ssd nil)
      (dup1 nil)   (a1 nil)     (a2 nil))
  (prog nil
      (setq ss1 (unknown-rule ss1))
      (setq ssd1 ss1)
      flag2
      (setq dup1 (car ssd1))
      (setq ssd1 (cdr ssd1))
      (setq ssd ssd1)
      (cond ((equal ssd1 nil) (go flag4)))
      flag3
      (cond ((equal ssd nil) (go flag2)))
      (cond ((and (equal (car dup1) (caar ssd)) (equal (cadr dup1)
(cadar ssd)))
            (setq dup2 (append dup2 (list (car ssd))))))
      (setq ssd (cdr ssd))
      (go flag3)
      flag4
      (setq a1 nil)
      (setq a2 (cddr ssd1))
      (setq ssd1 ss1)
      flag5
      (setq ssd (car ssd1))
      (cond ((equal (car dup2) ssd)
            (setq ss1 (append a1 a2)) (go flag4)))
      (setq a1 (append a1 (list ssd)))
      (setq a2 (cddr ssd1))
      (setq ssd1 (cdr ssd1))
      (cond ((equal ssd1 nil) (go flag6)))
      (go flag5)
      flag6
        (setq dup2 (cdr dup2))
      (cond ((equal dup2 nil) (setq ss2 ss1) (return)))
      (go flag4))))

(defun findunknown (ss1)
  (let ((dup2 nil)       (count1 0)   (ssd nil)    (ssd1 nil)
      (c2 nil)     (c1 nil)     (ss3 nil))
  (prog nil
      (setq dup2 nil)
      (setq uk nil)
      (setq count1 0)
      (setq ss2 nil)
      (setq ssd nil)
      (setq ssd1 ss1)
      (organize ss1)
    flag7
      (cond ((eql ss2 nil) (return)))
```

```
      (setq ss3 (car ss2))
      (setq c2 (append '(the) (list (car ss3)) '(of) (list (cadr
ss3)) '(is not.known)))
      (setq c1 (caar (kbcall 1 c2 1)))
      (cond ((equal c1 'not.known) (setq uk (append uk (list c2)))))
      (setq ss2 (cdr ss2))
      (cond ((equal ss2 nil) (setq uk (list uk)) (return)))
      (go flag7))))

COMMENT:  SIMULATION puts transient data into the qualitative plant
model

(defun simulation ()
  (let ((a 1)
        (old 0)
        (new 0)
        (normal 0)
        (xx nil))
  (setq a 0)
  (setq tstep (+ tstep 1))
  (prog nil
      label
         (if (= a ds) (return))
         (setq a (+ a 1))
         (setq old (aref attr a (- tstep 1)))
         (setq new (aref attr a tstep))
         (cond ((eql tstep 1)
                (kbcall 1 (append '(the) (list (aref cond1 a)) '(of)
(list (aref system a)) '(is) (append '(not.known) (list old))) 2)))
         (if (eql new nil) (return))
         (setq normal (aref attr a 0))
         (setq xx (list (append (vsnorm new normal) (list normal))))
         (cond ((eql (aref cond1 a) 'status)
               (if (> new 0.5) (setq xx '((on 1.0))))
               (if (< new 0.5) (setq xx '((off 0.0))))))
         (kbcall 1 (append '(the) (list (aref cond1 a)) '(of)
                           (list (aref system a)) '(is) xx) 2)
         (setq fl (aref cond1 a))
            (if (eql (aref cond1 a) 'pressure) (setq fl
'pressure.gradient))
            (if (eql (aref cond1 a) 'temperature) (setq fl
'temperature.gradient))
          (cond ((or (eql (aref cond1 a) 'pressure) (eql (aref cond1
a) 'temperature)
                     (eql (aref cond1 a) 'neutron.density))
                (setq xx (list (append (gradient old new) (list (- old
new)))))
                (kbcall 1 (append '(the) (list fl) '(of)
                                  (list (aref system a)) '(is) xx) 2)
                (cond ((eql (aref cond1 a) 'pressure)
                       (setq xx (list (append (ratedet old new) (list (-
old new)))))
                       (kbcall 1 (append '(the pressure.rate of)
                                         (list (aref system a)) '(is) xx)
2)))))
```

```
        (go label))))


(defun timint (aaa)
  (setq aaa (reverse aaa))
  (setq timl (+ (* (+ (* (digit-char-p (char aaa 4)) 10) (digit-char-
p (char aaa 3))) 60) (+ (* (digit-char-p (char aaa 1)) 10) (digit-
char-p (char aaa 0))))))


COMMENT:  INITVAL initializes variables

(defun initval ()
  (setq end nil)
  (let ((c nil))
      (setq rsys nil)
      (send *window2* :expose)
      (setq co 0)
      (setq cnn 0)
      (setq cn1 nil)
      (setq cn2 nil)
      (setq cn3 nil)
      (setq cno 0)
      (setq op 15.5)
      (setq orp 3800)
      (setq ot 650)
      (setq tlo 0)
      (setq tstep 0)
      (setq rslt nil)
      (setq graph 0)
      (setq transient (make-array 7))
      (send *window2* :line-out " ")
      (setq item-list nil)
      (prog nil
            (setq c 1)
            label
            (if (eql (aref system c) nil) (return))
            (setq item-list
                (append item-list
                        (list
                          (append (list
                                (string-append
                                  (prin1-to-string (aref system c)) "'s
"
                                  (prin1-to-string (aref cond1 c))))
                                (append '(:eval)
                                        (list (append '(grapher) (list
c))))))))
            (setq c (+ c 1))
            (go label))
      (send plotter-ops :set-item-list item-list)))

COMMENT:  PLOTTER-OPS is a basic pop up menu
```

```
(setq plotter-ops (tv:make-window
                'tv:pop-up-menu
                ':label "Plot Selection"
                ':borders 3
                ':item-list  '((one :eval (grapher 1))
                          (two :eval (grapher 2))
                          (three :eval (grapher 3))
                          (four :eval (grapher 4))
                          (five :eval (grapher 5))
                          (six :eval (grapher 6)))))

(setq p (make-array 8))

COMMENT:  RUN is the main body of the program.  It handles
communication between the inference engine and the ATMS knowledge
base

(defun run ()
  (let ((t1 nil)
       (t2 nil)
       (r 0)
       (c1 0)
       (len nil)
       (count nil)
       (count1 nil)
       (level 1)
       (check1 nil)
       (check  '((cool.sys)   (reactor.trip)   (steam.gen.sys)
(secondary.sys)
              (loss.of.offsite.power)))
       (case nil)
       (out nil)
       (tran (make-array 9))
       (prob (make-array 7)))
   (prog nil
       (if (equal time nil) (return))
       (initval)
       flag1
       (setq check1 check)
       (setq len 1)
       flag2
       (if (= len (+ (length check1) 1)) (go flag3))
       (setf (aref transient len)
             (append (car check)
                     (car (kbcall 2 (append '(the pattern of) (car
check) '(is ?A)) 1))))
       (setq len (+ len 1))
       (setq check (cdr check))
       (go flag2)
       flag3
       (setq ss1 nil)
       (setq c1 0)
       flag3.1
       (setq c1 (+ c1 1))
       (setq ss1 (append ss1 (cdr (aref transient c1))))
```

```
      (cond ((= c1 (length check1)) (findunknown ss1) (go begin)))
      (go flag3.1 )
      begin
      (prog nil
            (setq r 0)
            start
            (setq r (+ r 1))
            (cond ((= r (+ (length check1) 1)) (return)))
            (setf (aref prob r) 100)
            (setq t1 (cdr (aref transient r)))
            (setq case (car (aref transient r)))
            test
            (setq t2 (car t1))
            (setq t1 (cdr t1))
            (cond ((equal t2 nil) (go start)))
            (setf (aref prob r) (- (aref prob r) (car (unknown-search
t2))))
             (cond ((and (neq cn2 'sen) (eql (aref prob r) 0)) (setf
(aref prob r) 1)))
             (go test))
      (prog nil
            label
             (setq cn (aref time tstep))
             (cond ((eql cn nil) (go flag6)))
             (cond ((neq cn cno) (send *window2* :set-label (string-
append "          Transient Report at " (princ-to-string cn) "s "))
(setq cnn (- cn cno))))
             (setq cno cn)
             (setq count 0)
             (prog nil
                   loop
                    (cond ((= count (length check1)) (return)))
                    (setq count (+ count 1))
                    (setq test (cdr (aref transient count)))
                    (setq case  (car (aref transient count)))
                    (setf (aref p count) 1)
                    (prog nil
                        label1
                           (cond ((equal test nil)
                                  (if (> (aref prob count) 0)
                                      (setf (aref p count)
                                            (float (* (/ (aref p count)
(aref prob count)) 100))))
                                   (kbcall 3 (append '(the probability
of) (list case) '(is) (list (list (aref p count)))) 2)
                                     (setq  out  (append  (list  case)
'(probability is) (aref p count)))
                                   (return)))
                                (setf (aref p count) (+ (aref p count)
(eval-rule (car test))))
                           (setq test (cdr test))
                           (go label1))
                    (go loop))
              (setq count1 1)
              flag5
```

```
            (setq case (car check1))
            (cond ((> (aref p count1) 69)
                (setf (aref tran level) (append (aref tran level)
(list case)))
                (send *window2* :string-out (prin1-to-string case))
                (send *window2* :y-scroll-to 1 :relative-jump)
                (setq rsys (append rsys (cdr (aref transient
count1))))))
;;              (kbcall 3 (append '(the probability of) case '(is
(0.0))) 2)))
            (setq rslt (append rslt (list (append (list case) (list
(aref p count1)) (list level) (list (aref prob count1))))))
            (setq count1 (+ count1 1))
            (setq check1 (cdr check1))
            (cond ((> count1 count) (go flag51)))
            (go flag5)
            flag51
            (cond ((equal (aref tran level) nil) (go flag59)))
            (setq case (car (aref tran level)))
            (setf (aref tran level) (cdr (aref tran level)))
            (setq check (car (kbcall 2 (append '(the daughter of)
case '(is ?A)) 1)))
            (cond ((equal check '((nil))) (go flag59)))
            (setq level (+ level 1))
            (go flag1)
            flag59
            (cond ((and (= level 0) (neq (aref tran 1) nil)) (setq
level 1) (go flag51)))
            (cond ((= level 0) (setq level 1) (go flag6)))
            (setq level (- level 1))
            (go flag51)
            flag6
            (setq cn1 (test))
            (setq end nil)
            (if (eql cn1 'end) (return))
            (cond ((> graph 0) (grapher2 graph)))
            (cond ((not (eql cn nil)) (simulation) (send *window2*
:line-out " ")))
            (setq check '((cool.sys) (reactor.trip) (steam.gen.sys)
(secondary.sys) (loss.of.offsite.power)))
            (if (equal cn1 'ok) (display-result rslt))
            (setq rslt nil)
            (setq rsys nil)
            (go flag1)))
  (prim)))

COMMENT:  OPTIONS is the main menu used during a simulation run

(setq options (tv:make-window
                'tv:pop-up-menu
                ':label "Option Selection"
                ':borders 3
                ':item-list  '(("Gauge Report" :eval (gauge-report))
```

```
                                ("Deactivate Report Window(s)" :eval
(deactivate-window))
                                  ("Add Information" :eval (add-
information))
                                ("Condition Plots" :eval (cond-plot))
                                ("Diagnostic Result" :eval (setq end 'ok))
                                  ("Print Valued Results" :eval (print
rslt))
                                ("Uncertainty Screen" :eval (cond ((equal
cn2 'sen) (setq cn2 nil)) ((setq cn2 'sen))))
                                ("Exit" :eval (setq end 'end)))))

(defun test ()
  (tv:window-call (*mouse* :activate)
    (send *mouse* ':any-tyi-no-hang))
  (cond ((= co 0) (send *mouse* ':item ':new-type "Options Menu")
(setq co 1)))
  (prog nil
      (cond ((and (< tv:mouse-x 488) (> tv:mouse-x 127) (< tv:mouse-y
160) (> tv:mouse-y 50))
            (go label)))
      (return)
      label
      (send options ':expose)
      (send options ':choose)
      (send options ':bury)
      (return end)))

COMMENT: The following functions perform some of the options offered
during a simulation run

(defun deactivate-window ()
  (send *result* :deactivate)
  (setq cn2 nil)
  (send *window3* :deactivate))

(defun add-information ()
  (send *standard-output* :select)
  (prog nil
      (cond ((eql rsys nil) (return)))
      (send *window2* :deselect)
      (findunknown rsys)
      (print uk))
  (setq end 'new)
  (input-info ds))

(defun gauge-report ()
  (let ((count2 0)
      (rsys1 nil)
      (test1 nil)
      (app nil))
  (prog nil
      (cond ((eql rsys nil) (return)))
      (setq count2 0)
      (send *window3* :expose t)
```

```
        (send *window3* :clear-history)
      (send *window3* :select)
      (organize rsys)
      (setq rsys ss2)
      (setq rsys1 rsys)
      flag
      (cond ((= count2 (length rsys)) (return)))
      (setq test1 (car rsys1))
      (setq app (append '(the) (list (car test1)) '(of) (list (cadr
test1)) '(is )))
      (send *window3* :line-out
            (princ-to-string   (append app (kbcall 1 (append app
'(?a)) 1))))
      (setq rsys1 (cdr rsys1))
      (setq count2 (+ count2 1))
      (go flag))))

(defun cond-plot ()
  (send plotter-ops ':expose-near '(:mouse))
  (send plotter-ops ':choose)
  (send plotter-ops ':deactivate))

(defun make-item ()
  (let ((cnt 0)
      (blah nil)
      (strg nil))
  (prog nil
      (setq cnt 1)
      (setq blah '((fghj)))
      label
      (if (= cnt 30) (return))
      (setq strg (princ-to-string (list cnt)))
      (setq blah (append blah (list (tv:scroll-parse-item
                          "line number:   "
                          '(:string (princ-to-string (list cnt)))))))
      (setq cnt (+ cnt 1))
      (go label))))


(defun aaa ()
  (send *test* :set-display-item blah))


(defun comp ()
 (with-open-file (stream "kbs1:>bob>comp.dat" ':direction ':output)
   (send stream ':string-out (prinl-to-string (aref kb 1)))))


(defun quick-sort (x)
  (sort x #'(lambda (x y) (> (second x) (second y))))
  (sort x #'(lambda (x y) (> (third x) (third y)))))
```

```
(defun quick-sort-1 (list)
  (let ((zxcv (copy-list list)))
   (sort zxcv #'(lambda (x y)   (and (not (or (symbolp (first x))
(symbolp (first y)))) (< (first x) (first y)))))
   (sort zxcv #'(lambda (x y)   (and (not (or (symbolp (first x))
(symbolp (first y)))) (< (first x) (first y)))))))

(defun get-level (x l)
  (let ((new nil) (y (copy-list x)) (z (copy-list x)))
    (sort y #'(lambda (x y) (if (and (= (third x) l) (not (member+ x
new)))
                          (setq new (append new (list x))))
                    (= (third x) l)))
    (sort x #'(lambda (x y) (if (and (= (third y) l) (not (member+ y
new)))
                          (setq new (append new (list y))))
                    (= (third y) l)))
    (setq x z)
    new))

(defun best-result (x)
  (let ((new nil) (y (copy-list x)) (z (copy-list x)))
     (sort y #'(lambda (x y) (if (and (>= (second x) 70.0) (not
(member+ x new)))
                          (setq new (append new (list x))))
                    (>= (second x) 70.0)))
     (sort x #'(lambda (x y) (if (and (>= (second y) 70.0) (not
(member+ y new)))
                          (setq new (append new (list y))))
                    (>= (second y) 70.0)))
    (setq x z)
    new))

(defun display-result (rslt)
  (prog (dy level unc dtr)
      (setq dy (make-array 5))
      (setq res (make-array 5))
      (setq dtr (make-array 5))
      (setf (aref dy 1) "The most probable area(s) of failure is ")
      (setf (aref dy 2) "Evidence indicates there is a(n) ")
      (setf (aref dy 3) "Specifically, ")
      (send *result* :expose)
      (send *result* :select)
      (send *result* :clear-history)
      (send *result* :set-cursorpos 0 0)
      (setq rslt1 rslt)
      (setq level 1)
      (setf (aref res 1) (best-result (get-level rslt 1)))
      (setf (aref res 2) (best-result (get-level rslt 2)))
      (setf (aref res 3) (best-result (get-level rslt 3)))
      (prog (res2 res1 res3)
         label1
            (setf (aref dtr 1) nil)
```

```
            (if (and (= level 1) (equal (aref res 1) nil)) (return))
             (cond ((equal (aref res level) nil) (setq level (- level
1)) (go label1)))
            (setq res3 (aref res level))
          label2
            (setq res1 (first res3))
             (cond ((equal res1 nil) (setq level (- level 1)) (go
label1)))
            (setq res3 (cdr res3))
            (cond ((and (not (member+ (first res1) (aref dtr level)))
(neq (aref dtr level) nil))
                  (go label2)))
             (setq res2 (string-trim "()" (princ-to-string (first
res1))))
             (send *result* :line-out (string-append (aref dy level)
res2))
            (setq unc (uncertainty (fourth res1)))
            (send *result* :line-out (string-append "Uncertainty is "
unc))
            (setf (aref dtr (+ 1 level)) (car (kbcall 2 (append '(the
daughter of) (first res1) '(is ?a)) 1)))
            (setf (aref res level) (remove+ res1 (aref res level)))
              (cond ((and (equal (aref dtr (+ 1 level)) '((nil)))
(equal (aref res level) nil))
                    (setq level (- level 2))
                    (if (< level 0) (setq level 0))))
             (cond ((and (equal (aref dtr (+ 1 level)) '((nil))) (neq
(aref res level) nil))
                    (setq level (- level 1))))
            (setq level (+ level 1))
            (go label1)))
  (setq cn1 nil))

(defun remove+ (i1 i2)
    (let ((i3 i2) (i4 nil))
  (prog nil
     label
      (cond ((equal i1 (car i3)) (return (append i4 (cdr i3)))))
      (setq i4 (append i4 (list (car i3))))
      (setq i3 (cdr i3))
      (if (equal i3 nil) (return nil))
      (go label))))

(defun member+ (i1 i2)
  (let ((i3 i2))
  (prog nil
     label
      (if (equal i1 (car i3)) (return t))
      (setq i3 (cdr i3))
      (if (equal i3 nil) (return nil))
      (go label))))

COMMENT:  UNCERTAINTY defines uncertainty of an event

(defun uncertainty (x)
```

```
  (prog nil
       (if (not (numberp x)) (return nil))
       (if (>= x 90) (return "very.low"))
       (if (>= x 70) (return "low"))
       (if (> x 30) (return "moderate"))
       (if (> x 10) (return "high"))
       (return "very.high")))

COMMENT:  UNKNOWN-SEARCH, EVAL-RULE, EVAL-CD, AND RULE perform the
actual processing of rules (inference engine)

(defun unknown-search (rule)
  (prog nil
       (eval-rule rule)
       (if (equal nk 'yes) (return (last rule)))
       (return '(0))))

(defun eval-rule (rule)
    (prog (len inq cd)
          (setq col nil)
          (setq nk nil)
          (setq len (length rule))
          (cond ((> len 2)
                 (setq inq (first (car (kbcall 1 (append '(the) (list
(first rule)) '(of) (list (second rule)) '(is ?a)) 1))))
                 (if (equal inq (third rule)) (return (fourth rule)))
                 (if (and (neq inq 'not.known) (neq inq 'normal) (neq inq
'nil) (equal (third rule) 'not.normal)) (return (fourth rule)))
                 (if (equal inq 'not.known) (setq nk 'yes))
                 (return '0)))
          (setq cd (first rule))
          (if (eval-cd cd) (return (second rule)))
          (return '0)))

(defun eval-cd (cd)
  (prog (cj cdl el inq exp res)
       (setq cj (first cd))
       (setq cdl (cdr cd))
       (setq el (list cj))
      label
       (cond ((equal cdl nil) (return (eval el))))
       (setq exp (car cdl))
       (cond ((conjunct? exp) (setq res (eval-cd exp)) (go label1)))
       (cond ((equal cj 'rule) (setq el '(and)) (setq res (rule exp))
(go label1)))
       (setq inq (first (car (kbcall 1 (append '(the) (list (first
exp)) '(of) (list (second exp)) '(is ?a)) 1))))
       (setq col (append col (list exp)))
       (cond ((or (equal inq (third exp))
                  (and (neq inq 'not.known)
                       (neq inq 'normal)
                       (neq inq 'nil)
                       (equal (third exp) 'not.normal)))
              (setq res 't))
             ((setq res 'nil)))
```

```
      (cond ((equal inq 'not.known) (setq nk 'yes)))
     label1
      (if (eq nk 'yes) (return nil))
      (setq el (append el (list res)))
      (setq cd1 (cdr cd1))
      (go label1)))

(defun conjunct? (exp)
  (prog nil
      (if (equal (first exp) 'and) (return 't))
      (if (equal (first exp) 'or) (return 't))
      (if (equal (first exp) 'rule) (return 't))
      (if (equal (first exp) 'not) (return 't))))

(defun rule (exp)
  (prog (c3 c4 kbn)
      (setq kbn 1)
      (if (equal (first exp) 'probability) (setq kbn 3))
      (setq c3
            (car
            (kbcall kbn
                  (append '(the) (list (first exp)) '(of) (list
(second exp)) '(is ?A)) 1)))
      (if (eql c3 nil) (return))
      (cond ((equal kbn 3) (setq c3 (append '(probability) c3))))
      (cond ((equal (fourth exp) 'number) (setq c4 (append '(number)
(list (fifth exp)))) (go label1)))
      (setq c4
            (car
            (kbcall kbn
                  (append '(the) (list (fourth exp)) '(of) (list
(fifth exp)) '(is ?A)) 1)))
     label
      (cond ((or (eql (first c3) 'not.known) (eql (first c4)
'not.known)) (setq nk 'yes) (return 'nil)))
      (setq c3 (second c3)) (setq c4 (second c4))
      (cond ((equal (third exp) '<) (go flag1)))
      (cond ((equal (third exp) '>) (go flag2)))
      (cond ((equal (third exp) '=) (go flag3)))
      (cond ((equal (third exp) '>=) (go flag4)))
      (cond ((equal (third exp) '<=) (go flag5)))
      (return 'nil)
     flag1
      (cond ((< c3 c4) (return 't)))
      (go flag6)
     flag2
  (cond ((> c3 c4) (return 't)))
  (go flag6)
  flag3
  (cond ((= c3 c4) (return 't)))
  (go flag6)
  flag4
  (cond ((>= c3 c4) (return 't)))
  (go flag6)
  flag5
```

```
  (cond ((<= c3 c4) (return 't)))
  flag6
  (return 'nil)))

COMMENT:  INPUT-INFO is the user interface that allows a user to tell
the computer what simulation data files to read

;;;  for initialization of a system


(defun input-info (ii)
  (let ((c 0)      (a 0) (kb1 nil)    (kb2 nil)    (kb3 nil)
       (fname nil) (sys nil)    (blip nil)   (item-list nil)
       (ans nil)   (ques nil)        (ans1 nil)        (ques1 nil)  (it
nil))
  (setq c 0)
  (setq a 0)
  (setq ds ii)
  (cond ((= ii 0) (setq time (make-array 800))
       (setq attr (make-array '(10 800)))
       (setq system (make-array 10))
       (setq cond1 (make-array 10))))
  (setq kb1 (aref kb 1))
  (send *window1* :expose)
  (send *window1* :clear-history)
  (prog nil
      label
         (setq kb1 (cdr kb1))
         (setq kb2 (caar kb1))
         (if (eql kb2 nil) (return))
         (send *window1* :set-cursorpos c a)
         (send *window1* ':item ':new-type (prin1-to-string kb2))
         (setq c (+ c 160))
         (cond ((> c 330) (setq c 0) (setq a (+ a 30))))
         (go label))
  (tv:window-call (*window1* :activate)
    (send *window1* :select)
    (prog nil
        label
           (setq kb1 (aref kb 1))
           (clean-blip)
           (setq blip (send *window1* :any-tyi))
           (setq sys (third blip))
           (if (not (listp blip)) (go label))
             (cond ((equal sys "END") (send *window1* :deactivate)
(return)))
           (select-info kb1 sys)
           (if (equal cond nil) (go label))
           (send *standard-output* :select)
           (cond ((y-or-n-p "Is the information from a datafile")
                (setq fname (string-append "sim>"
                                    (prompt-and-read :string "Enter
filename >")))
```

```
                (setq it (check-others (string-symbol sys) (string-
symbol cond)))
                (print it)
                (cond ((numberp it) (reader fname it) (go label)))
                (setq ds (+ ds 1))
                (setf (aref system ds) (string-symbol sys))
                (setf (aref cond1 ds) (string-symbol cond))
                (reader fname ds)
                (print 'Done)
                (go label)))
          (setq ques (string-append "The " cond " of " sys " is ?"))
          (setq ques1 (string-append "Its value is ?"))
          (setq sys (string-symbol sys))
          (setq cond (string-symbol cond))
          (setq ans (prompt-and-read :symbol ques))
             (setq ans1 (string-symbol (prompt-and-read :string
ques1)))
            (cond ((not (numberp ans1)) (setq ans1 (second (car
(kbcall 1 (append '(the) (list cond) '(of) (list sys) '(is ?a))
1))))))
          (setq ans (append (list ans) (list ans1)))
            (kbcall 1 (append '(the) (list cond) '(of) (list sys)
'(is) (list ans)) 2)
          (go label))
    (cond ((= ii 0) (prim))))))

(defun check-others (sys cond)
  (prog (c)
      (setq c 0)
     label
      (if (= c ds) (return nil))
      (setq c (+ c 1))
      (cond ((and (equal (aref system c) sys) (equal (aref cond1 c)
cond)) (return c)))
      (go label)))

(defun select-info (kb1 sys)
  (prog nil
     label
      (setq kb1 (cdr kb1))
      (setq kb2 (prin1-to-string (caar kb1)))
      (if (equal kb2 "NIL") (return))
      (cond ((equal kb2 sys)
             (setq c 0)
             (setq kb3 (cadar kb1))
             (setq item-list nil)
             (prog nil
                 (setq c 1)
              label
                 (if (equal (caar kb3) 'adj.sys) (return))
                 (if (equal (caar kb3) nil) (return))
                 (setq item-list
                     (append item-list
                           (list (append (list (prin1-to-string (caar
kb3)))
```

```
                                (append '(:eval) (list (append '(setq
cond)
                                                       (list (prin1-to-string (caar
kb3)))))))))))
                    (setq c (+ c 1))
                    (setq kb3 (cdr kb3))
                    (go label))
                (setq cond nil)
                (send II-pop-up :set-item-list item-list)
                (send II-pop-up ':expose-near '(:mouse))
                (send II-pop-up ':choose)
                (send II-pop-up ':deactivate)))
        (go label)))

(defun clean-blip ()
  (let ((blip nil)
        (blip1 nil))
    (tv:window-call (*window1* :activate)
      (prog nil
          label
              (setq blip1 (send *window1* :any-tyi-no-hang))
              (if (and (eql blip1 nil) (eql blip nil)) (return))
              (print blip)
              (go label)))))

(defun string-symbol (str)
  (prog (str1 str2 str3 exp1 exp2 exp)
        (if (equal str "") (return))
        (cond ((not (equal (string-search " " str) nil)) (return)))
        (setq str (string-upcase str))
        (setq str1 (intern str))
        (setq str2 (prin1-to-string str1))
        (cond ((equal (substring str2 0 1) "|")
               (setq exp1 (string-search "." str))
               (prog nil
                   (cond ((equal exp1 nil) (setq exp1 0) (return)))
                   (setq str (string-append (string-trim "." (substring
str 0 (+ exp1 1))) (substring str (+ exp1 1) (length str))))
                label
                   (setq exp1 (- (length str) exp1)))
               (setq exp2 (string-search "E" str))
               (prog nil
                   (cond ((equal exp2 nil)
                          (setq exp (- 0 exp1))
                            (cond ((= exp 0) (setq str1 (parse-integer
str)) (return)))
                          (go label)))
                   (setq exp1 (- exp1 (- (length str) exp2)))
                   (setq str3 (substring str 0 exp2))
                   (setq exp2 (parse-integer (substring str (+ exp2 1)
(length str))))
                   (setq str str3)
                   (setq exp (- exp2 exp1))
               label
```

```
                (setq str1 (float (* (parse-integer str) (expt 10
exp)))))))
      (return str1)))

(defun end ()
  (send large-win :bury))


COMMENT:  GRAPHER, GRAPHER2, MAXMIN, and FIX are used to develop
graphs of paired datasets

(defun grapher (i)
  (let ((t1 1)     (max nil)    (min nil)    (cdn1 nil)   (t2 nil)
       (inc nil)   (incl nil)  (c 0) (spac 0))
;;; find maximum and minimums
  (setq t1 1)
  (setq max (aref attr i 0))
  (setq min max)
  (prog nil
      label
         (cond ((= t1 tstep) (return)))
          (cond ((> (aref attr i t1) max) (setq max (aref attr i
t1))))
          (cond ((< (aref attr i t1) min) (setq min (aref attr i
t1))))
         (setq t1 (+ t1 1))
         (go label))
  (setq ymax (* (round (+ max 1))))
  (setq ymin (* (round (- min 1))))
  (cond ((> (- ymax ymin) 100) (setq ymin 0)))
  (prog nil
      (setq t1 0)
      label
      (if (eql (aref time t1) nil) (return))
      (setq xmax (aref time t1))
      (setq t1 (+ t1 1))
      (go label))
;;; set-up
  (send *window* :set-label
      (string-append
        (prin1-to-string
          (aref system i)) "'s " (prin1-to-string (aref cond1 i)) "
vs. time (s)"))
  (setq graph i)
  (setq t2 0)
  (setq ymax (- ymax ymin))
  (prog nil
  flag
  (setq inc (float (* (/ (round (- (/ ymax 120) 0.5)) 2) 60)))
   (cond ((= inc 0) (setq ymax (* ymax 10)) (setq t2 (+ t2 1)) (go
flag)))
  (cond ((> t2 0) (setq inc (/ inc (expt 10 t2))) (setq ymax (/ ymax
(expt 10 t2))) )))
  (setq incl (* (round (* 240 (/ inc ymax)))))
  (send *window* :expose)
```

```
  (send *window* :clear-history)
  (send *window* :set-cursorpos 40 600)
  (send *window* :string-out "Graph")
  (send *window* :draw-line 60 20 60 260)
  (send *window* :set-cursorpos 20 255)
  (send *window* :string-out (prin1-to-string ymin))
  (send *window* :set-cursorpos 20 (- 255 incl))
  (send *window* :string-out (prin1-to-string (fix (+ inc ymin) 2)))
  (send *window* :set-cursorpos 20 (- 255 (* incl 2)))
  (send *window* :string-out (prin1-to-string (fix (+ (* inc 2) ymin)
2)))
  (send *window* :set-cursorpos 20 (- 255 (* incl 3)))
  (send *window* :string-out (prin1-to-string (fix (+ (* inc 3) ymin)
2)))
  (send *window* :set-cursorpos 20 (- 255 (* incl 4)))
  (send *window* :string-out (prin1-to-string (fix (+ (* inc 4) ymin)
2)))
  (send *window* :draw-line 60 260 580 260)
  (cond ((> (- 255 (* incl 5)) 20)
        (send *window* :set-cursorpos 20 (- 250 (* incl 5)))
        (send *window* :string-out (prin1-to-string (fix (+ (* inc 5)
ymin) 2)))))
  (cond ((> (- 255 (* incl 6)) 20)
        (send *window* :set-cursorpos 20 (- 250 (* incl 6)))
        (send *window* :string-out (prin1-to-string (fix (+ (* inc 6)
ymin) 2)))))
  (send *window* :set-cursorpos 55 265)
  (prog nil
       (setq c 0)
       (setq spac (truncate (/ (+ 10 xmax) 10)))
       label
       (if (> (* c spac) xmax) (return))
       (send *window* :set-cursorpos (+ 55 (* c 52)) 265)
       (send *window* :string-out (prin1-to-string (* c spac)))
       (setq c (+ c 1))
       (go label))
  (setq tlo 0)
;;;  graph
  (grapher2 i)))

(defun grapher2 (i)
  (let ((x 0)
       (y 0)
       (tl 0))
  (setq tl tlo)
  (prog nil
       label
       (cond ((= tl tstep) (return)))
       (setq y (* (round (- 260 (/ (* 240 (- (aref attr i tl) ymin))
ymax)))))
       (setq x (* (round (+ 60 (* (aref time tl) (/ 520 xmax))))))
       (cond ((= tl 0) (go label1)))
       (send *window* :draw-line xold yold x y)
       label1
       (setq xold x)
```

```
      (setq yold y)
      (setq t1 (+ t1 1))
      (go label))
  (setq t1o tstep)))


(defun maxmin (i)
  (setq t1 1)
  (setq max (aref attr i 0))
  (setq min max)
  (prog nil
      label
         (cond ((= t1 tstep) (return)))
         (cond ((> (aref attr i t1) max) (setq max (aref attr i
t1))))
         (cond ((< (aref attr i t1) min) (setq min (aref attr i
t1))))
         (setq t1 (+ t1 1))
         (go label))
  (setq ymax (* (round (+ max 1))))
  (setq ymin (* (round (- min 1))))
  (cond ((> (- ymax ymin) 100) (setq ymin 0)))
  (prog nil
      (setq t1 0)
      label
      (if (eql (aref time t1) nil) (return))
      (setq xmax (aref time t1))
      (setq t1 (+ t1 1))
      (go label)))


(defun fix (num dp)
   (setq num (float (/ (* (round (* num (expt 10 dp)))) (expt 10
dp)))))

COMMENT: TREE, TREE-INFO, and DIS-TREE-INFO are used to create the
mouse sensitive tree display of the ATMS network nodes

(defun tree (par)
  (let ((y 0)      (ymax 0)   (n 0)    (k (make-array 7))    (kl (make-
array 7))
      (l (make-array 7)) (tp1 0) (tp2 0) (i nil) (h 0) (pos (make-
array '(10 10 10)))
      (pre 0)      (past 0))
  (prog nil
      (send large-win :refresh)
      (send large-win :expose)
      (send large-win :set-cursorpos 0 0)
      (send large-win ':item ':exam "Examine")
      (setq y 100)
      (setq ymax 100)
      (setq n 1)
      (setf (aref k 1) 4)
      (setf (aref k1 1) 0)
      (setf (aref l 1) par)
```

```
      (setq tp1 0)
      (setq pos (make-array '(10 10 10)))
      loop
      (prog nil
         loop1
            (cond ((eql (aref l n) nil)
                   (setq y (- y (/ (* (aref k n) h) 3)))
                   (setq h (+ 20 (- ymax y)))
                   (setq y (- y h))
                   (setq n (- n 1))
                   (return)))
            (setq i (car (aref l n)))
            (setq n (+ n 1))
             (setf (aref l n) (car (kbcall 2 (append '(the daughter
of) i '(is ?A)) 1)))
            (cond ((equal (aref l n) '((nil))) (setq h 15) (setq n (-
n 1)) (return)))
            (setf (aref k1 n) 0)
            (setf (aref k n) (length (aref l n)))
            (go loop1))
      (cond ((= n 0) (send large-win :set-cursorpos 0 0) (return)))
      (setf (aref k1 n) (+ (aref k1 n) 1))
      (setq pre (aref k1 n))
      (setq past (aref k1 (- n 1)))
      (if (eql past nil) (setq past 0))
      (setq y (+ y h))
      (setq i (car (aref l n)))
      (setposition n y i)
      (setf (aref pos n past pre) (round (+ y 5)))
      (setq tp1 (- (aref k1 n) 1))
      (setq tp2 1)
      loop2
      (cond ((not (eql (aref pos (+ n 1) tp1 tp2) nil))
                  (send large-win :draw-line (* 150 (+ n 1))
                      (aref pos (+ n 1) tp1 tp2) (- (* 150 (+ n 1))
30) (aref pos n past pre))
                  (setf (aref pos (+ n 1) tp1 tp2) nil)
                  (setq tp2 (+ tp2 1))
                  (go loop2)))
      (if (> y ymax) (setq ymax y))
      (setq y ymax)
      (setf (aref l n) (cdr (aref l n)))
      (go loop)))
  (send large-win :set-cursorpos 0 800)
  (prim))


(defun setposition (n y i)
  (setq y (round y))
  (send large-win :set-cursorpos (* 150 n) y)
  (send large-win ':item ':new-type (prin1-to-string (car i))))
;;  (send large-win :string-out (princ-to-string (car i))))


(defun tree-info ()
```

```
  (send large-win :expose)
  (send large-win :select)
  (tv:window-call (large-win :activate)
    (send large-win :select)
    (prog (blip sys ilist)
      label
       (clean-blip2)
       (setq blip (send large-win :any-tyi))
       (if (not (listp blip)) (go label))
       (setq sys (third blip))
       (cond ((equal sys "END") (return)))
       (if (not (stringp sys)) (go label))
       (setq sys (list (string-symbol sys)))
       (setq ilist (car (kbcall 2 (append '(the pattern of) sys '(is
?a)) 1)))
       (prog (c c1 get)
           (setq c ilist)
           (setq ilist nil)
          label
           (if (equal c nil) (return))
           (setq c1 (princ-to-string (car c)))
           (cond ((> (length c1) 50)
                  (setq c2 (string-search "
" c1))
                      (if (neq c2 nil) (setq c1 (string-append
(substring c1 0 (- c2 1)) (substring c1 (+ c2 1) (length c1)))))
                      (setq c1 (string-append (substring c1 0 49)
"...")))))
           (setq get (append (list c1)
                             (append '(:eval) (list (append '(dis-tree-
info) (list (append '(quote) (list (car c))))))))))
           (setq ilist (append ilist (list get)))
           (setq c (cdr c))
           (go label))
       (send II-pop-up :set-item-list ilist)
       (send II-pop-up ':expose-near '(:mouse))
       (send II-pop-up ':choose)
;;     (send large-win :any-tyi)
       (send II-pop-up ':deactivate)
       (cond ((equal cn3 'off) (setq cn3 'on) (return)))
       (go label))))

(defun dis-tree-info (c)
  (prog (r)
      (if (equal c '(nil)) (return))
      (send large-win :set-cursorpos 0 30)
      (send large-win :clear-rest-of-line)
      (cond ((> (eval-rule c) 0) (setq r "True")) ((setq r "False")))
      (if (equal nk 'yes) (setq r "Not.known"))
      (send large-win :string-out r)))

(setq l (make-array 7))
(setq k (make-array 7))
(setq kl (make-array 7))
```

```
(defun clean-blip2 ()
  (let ((blip nil)
        (blip1 nil))
    (prog nil
       label
        (setq blip1 (send large-win :any-tyi-no-hang))
        (if (and (eql blip1 nil) (eql blip nil)) (return))
        (go label))))
```

COMMENT: DEVELOP-SIMULATION is used to reproduce graphs from journals reporting output from thermal-hydraulic codes.

```
(defun develop-simulation ()
  (prog nil
 redo
    (send *standard-output* :select)
    (let*  ((stop 0)
       (dataset (make-array 800))
       (x-mouse (make-array 80))
       (y-mouse (make-array 80))
       (mouse-x nil)
       (mouse-y nil)
        (cdn1  (prompt-and-read  :symbol  "Enter  the  name  of  the
condition you are defining >"))
       (fname (string-append "Nukes:>catalisp>sim>" (prompt-and-read
:string "Enter filename (Nukes:>catalisp>sim>) >")))
        (step  (prompt-and-read  :number  "Enter  interval  spacing
(seconds) >"))
       (ymin (prompt-and-read :number "Enter lower boundary >"))
       (ymax  (-  (prompt-and-read  :number  "Enter  higher  boundary  >")
ymin))
       (xmax (prompt-and-read :number "Enter length of time >"))
       (mouse-y-old (- (prompt-and-read :number (string-append "Enter
" (prin1-to-string cdn1) " at time zero >")) ymin))
       (t2 0)
       (inc 0)
       (inc1 0)
       (mouse-x-old 60)
       (spac 0)
       (c 0)
       (t1o 0)
       (mouse-x 0)
       (mouse-y 0)
       (mouse-char nil)
       (blip nil)
       (aa nil)
       (new-y 0)
       (slope 0)
       (c1 0)
       (c2 0)
       (c3 0)
       (c4 0))
  (if (not (y-or-n-p "Is this OK?")) (go redo))
```

```
  (send *window4* :set-label (string-append (prin1-to-string cdn1) "
vs. time (s)"))
  (prog nil
  flag
  (setq inc (float (* (/ (round (- (/ ymax 120) 0.5)) 2) 60)))
   (cond ((= inc 0) (setq ymax (* ymax 10)) (setq t2 (+ t2 1)) (go
flag)))
  (cond ((> t2 0) (setq inc (/ inc (expt 10 t2))) (setq ymax (/ ymax
(expt 10 t2))) )))
  (setq incl (* (round (* 240 (/ inc ymax))))))
  (setq mouse-y-old (truncate (- 275 (* incl (/ mouse-y-old inc)))))
  (setq mouse-x-old 60)
  (send *window4* :expose)
  (send *window4* :clear-history)
  (send *window4* :set-cursorpos 40 620)
  (send *window4* :string-out "Graph")
  (send *window4* :draw-line 60 40 60 280)
  (send *window4* :set-cursorpos 20 270)
  (send *window4* :string-out (prin1-to-string ymin))
  (send *window4* :set-cursorpos 20 (- 270 incl))
  (send *window4* :string-out (prin1-to-string (fix (+ inc ymin) 2)))
  (send *window4* :set-cursorpos 20 (- 270 (* incl 2)))
   (send *window4* :string-out (prin1-to-string (fix (+ (* inc 2)
ymin) 2)))
  (send *window4* :set-cursorpos 20 (- 270 (* incl 3)))
   (send *window4* :string-out (prin1-to-string (fix (+ (* inc 3)
ymin) 2)))
  (send *window4* :set-cursorpos 20 (- 270 (* incl 4)))
   (send *window4* :string-out (prin1-to-string (fix (+ (* inc 4)
ymin) 2)))
  (send *window4* :draw-line 60 280 580 280)
  (cond ((> (- 275 (* incl 5)) 20)
       (send *window4* :set-cursorpos 20 (- 270 (* incl 5)))
       (send *window4* :string-out (prin1-to-string (fix (+ (* inc 5)
ymin) 2)))))
  (cond ((> (- 275 (* incl 6)) 20)
       (send *window4* :set-cursorpos 20 (- 270 (* incl 6)))
       (send *window4* :string-out (prin1-to-string (fix (+ (* inc 6)
ymin) 2)))))
  (send *window4* :set-cursorpos 55 285)
  (prog nil
      (setq c 0)
      (setq spac (truncate (/ xmax 10)))
      label
      (if (> (* c spac) xmax) (return))
      (send *window4* :set-cursorpos (+ 55 (* c 52)) 285)
      (send *window4* :string-out (prin1-to-string (* c spac)))
      (setq c (+ c 1))
      (go label))
  (send *window4* :draw-filled-in-circle mouse-x-old mouse-y-old 2)
  (setf (aref x-mouse 0) mouse-x-old)
  (setf (aref y-mouse 0) mouse-y-old)
  (setq t1o 0)
  (send *window4* :set-mouse-position 170 170)
  (setq c 1)
```

```
(prog nil
     label
     (cond ((eql stop 1) (setq stop nil) (return)))
     (tv:window-call (*window4* :activate)
       (send *window4* :select)
       (prog nil
          label
           (setq blip (send *window4* :any-tyi))
           (send *window4* :set-cursorpos 0 0)
           (send *window4* ':item ':new-type "Completed")
           (send *window4* :set-cursorpos 0 20)
           (send *window4* ':item ':new-type "Undo")
           (if (not (listp blip)) (go label))
           (cond ((equal (third blip) "Completed")
                  (setq stop 1) (return)))
           (cond ((equal (third blip) "Undo") (setq c (- c 1))
                  (cond ((<= c 0) (setq c (+ c 1)) (return)))
                   (send *window4* :draw-line (aref x-mouse c) (aref
y-mouse c) (aref x-mouse (- c 1)) (aref y-mouse (- c 1)) tv:alu-
andca)
                    (send *window4* :draw-filled-in-circle (aref x-
mouse c) (aref y-mouse c) 2 tv:alu-andca)
                  (setq mouse-x-old (aref x-mouse (- c 1)))
                  (setq mouse-y-old (aref y-mouse (- c 1)))
                  (setq mouse-x (aref x-mouse (- c 1)))
                  (setq mouse-y (aref y-mouse (- c 1)))
                  (return)))
           (setq aa (substring (prin1-to-string (third blip)) 2 11))
           (setq mouse-x (fourth blip))
           (setq mouse-y (fifth blip))
           (if (or (equal mouse-x nil) (equal mouse-y nil)) (go
label))
           (cond ((equal aa "SCROLLER2") (go label1)))
           (go label)
         label1
           (setf (aref x-mouse c) mouse-x)
           (setf (aref y-mouse c) mouse-y)
           (setq mouse-char (second blip))
           (cond ((char-mouse-equal mouse-char #\mouse-r)
                  (send *window4* :set-cursorpos 100 0)
                  (send *window4* :clear-rest-of-line)
                   (setq c3 (truncate (/ (- (aref x-mouse c) 60) (/
520 xmax))))
                   (setq c4 (+ ymin (* inc (/ (* -1 (- (aref y-mouse
c) 275)) incl))))
                  (send *window4* :string-out (string-append (prin1-
to-string c3) " "(prin1-to-string (fix c4 3))))
                  (send *window4* :set-cursorpos 450 0)
                  (send *window4* :string-out (string-append (prin1-
to-string c3) " "(prin1-to-string (fix c4 3))))
                  (go label)))))
     (if (<= mouse-x mouse-x-old) (go label))
     (send *window4* :draw-filled-in-circle mouse-x mouse-y 2)
     (send *window4* :draw-line mouse-x-old mouse-y-old mouse-x
mouse-y)
```

```
      (setq mouse-x-old (aref x-mouse c))
      (setq mouse-y-old (aref y-mouse c))
      (setq c (+ c 1))
      (go label))
   (setq stop 0)
   (setq c 0)
   (setq c1 0)
   (prog nil
      (setq c2 0)
     label
      (if (eql (aref x-mouse c2) nil) (return))
      (setf (aref x-mouse c2) (truncate (/ (- (aref x-mouse c2) 60)
(/ 520 xmax))))
      (setf (aref y-mouse c2) (+ ymin (* inc (/ (* -1 (- (aref y-
mouse c2) 275)) inc1))))
      (setq c2 (+ c2 1))
      (go label))
   (setq c3 xmax)
   (setq c4 0)
   (prog nil
      label
         (cond ((> c1 xmax) (setq stop 0) (return)))
         (prog nil
            (setq c2 0)
            label1
            (cond ((eql (aref x-mouse (+ c2 1)) nil) (setq c3 (aref
x-mouse c2))
                  (setq c2 (- c2 1))
                  (setq c4 c2)))
            (cond ((or (and (>= c1 (aref x-mouse c2)) (<= c1 (aref
x-mouse (+ c2 1))))
                       (and (>= c1 c3) (<= c1 xmax)))
                  (if (>= c1 c3) (setq c2 c4))
                  (setq slope (/ (- (aref y-mouse (+ c2 1)) (aref y-
mouse c2)) (- (aref x-mouse (+ c2 1)) (aref x-mouse c2))))
                  (setq new-y (+ (* (- c1 (aref x-mouse c2)) slope)
(aref y-mouse c2)))
                  (setf (aref dataset c) (string-append (prin1-to-
string (float c1)) "     " (prin1-to-string new-y)))
                  (return)))
            (setq c2 (+ c2 1))
            (go label1))
         (setq c1 (+ c1 step))
         (setq c (+ c 1))
         (go label))
   (with-open-file (stream fname
                    ':byte-size nil
                    ':characters t
                    ':direction ':output)
      (setq c 0)
      (prog nil
         label
            (if (eql (aref dataset c) nil) (return))
            (send stream ':line-out (aref dataset c))
            (setq c (+ c 1))
```

```
          (go label))))
  (prim)))

COMMENT:  Below are define the windows used in CATALisp

(defflavor scroller ()
         (dw:dynamic-window
          dw:margin-ragged-borders
          tv:text-scroll-window))

(defflavor scroller1 ()
       (dw:dynamic-window
        dw:margin-ragged-borders))

(defflavor new ()
          (tv:centered-label-mixin
           tv:borders-mixin tv:top-box-label-mixin
           tv:basic-mouse-sensitive-items
           tv:window
           tv:sheet))

(setq default 0)
(defvar alist-alpha nil)
(defvar alist-beta nil)


(defflavor scroller2 ()
       (tv:basic-mouse-sensitive-items
       dw:dynamic-window))

(defflavor scroller3 ()
       (tv:basic-mouse-sensitive-items
        tv:typeout-window
        dw:dynamic-window))

(setq alist-alpha '((:new-type nil "Right: End"
                        ("  End  " :value nil :documentation "
Nil"))))

(setq alist-beta '((:new-type nil "Right: End"
                        ("  End  " :eval (setq cn3 'off)
:documentation "End Tree Info"))
                (:exam nil "Right: Menu of Tree Functions"
                     ("  Tree-Info  " :eval (tree-info) :documentation
"  Tree Info  ")
                     ("     End     " :eval (setq cn3 'off)
:documentation "  End  "))))


(defvar *window1* (tv:make-window 'scroller2
                                  ':x 630
                                  ':y 50
```

```
                              ':width 500
                              ':height 370
                              ':default-character-style '(:dutch
:bold :small)
                              ':item-type-alist alist-alpha))

(defvar dynwin  (tv:make-window 'scroller1
                        :x 730
                        :y 492
                        :width 390
                        :height 300
                        :default-character-style '(:jess :bold
:normal)
                        :expose-p nil
                        :end-of-page-mode :truncate
                        :margin-components
                        '((dw:margin-borders :thickness 1)
                        (dw:margin-white-borders :thickness 3)
                        (dw:margin-borders :thickness 10)
                        (dw:margin-white-borders :thickness 8)
                        (dw:margin-borders :thickness 3)
                        (dw:margin-whitespace :margin :left
:thickness 10)
                        (dw:margin-scroll-bar)
                        (dw:margin-whitespace :margin :bottom
:thickness 7)
                        (dw:margin-scroll-bar :margin :top)
                        (dw:margin-whitespace :margin :left
:thickness 10)
                        (dw:margin-label :margin :bottom
                                  :style (:sans-serif :italic
:normal))
                        (dw:margin-whitespace :margin :top
:thickness 10)
                        (dw:margin-whitespace :margin :right
:thickness 13))
))

(defvar *window* (tv:make-window 'scroller1
                              :x 64
                              :y 441
                              :width 359
                              :height 350
                              :default-character-style '(:dutch :bold
:small)
                              :end-of-page-mode :truncate
                              :name "        RCS pressure (psia) vs
time (sec)"))



(defvar *window2* (tv:make-window 'dw:dynamic-window
                        :x 530
                        :y 44
```

```
                        :width 600
                        :height 120
                        :name "                    Transient Report"
                        :end-of-page-mode :scroll
                        :scroll-factor 100
                        :default-character-style '(:swiss :bold
:small)))

(defvar *window3* (tv:make-window 'scroller1
                        :x 64
                        :y 170
                        :height 270
                        :width 450
                        :end-of-page-mode :scroll
                        :scroll-factor 100
                        :name "Gauge Report"
                        :default-character-style '(:swiss :roman
:small)
                        ))

(defvar *result* (tv:make-window 'scroller1
                        :x 524
                        :y 170
                        :height 270
                        :width 450
                        :end-of-page-mode :scroll
                        :scroll-factor 100
                        :name "Diagnostic Report"
                        :default-character-style '(:swiss :roman
:small)
                        ))

(defvar *mouse*
      (tv:make-window
        'new
        ':borders 2
        ':top 50
        ':bottom 160
        ':right 488
        ':width 316
        ':blinker-p nil
        ':label '(:string "OPTIONS" :font fonts:bigfnt)
        ':default-character-style '(:swiss :roman :normal)
        ))


(setq II-pop-up (tv:make-window
                'tv:momentary-menu
                ':label "Plot Selection"
                ':borders 3
                ':item-list  '("Nothing")))

(setq primary (tv:make-window
                'tv:menu
                ':x 64
```

```
                ':y 43
                ':label '(:string "   CATALisp" :font fonts:bigfnt)
                ':borders 3
                ':item-list  '(("Develop a Simulation" :eval
(develop-simulation))
                               ("Input a New Simulation" :eval (input-
info 0))
                               ("Run the Simulation" :eval (run))
                               ("Display Tree Structure" :eval (tree
'((not.normal))))
                               ("Exit" :eval (end)))))

(setq large-win (tv:make-window 'scroller2
                                :x 64
                                :y 163
                                :width 1067
                                :height 610
                                :default-character-style '(:swiss :bold
:small)
                                ':item-type-alist alist-beta))


(defvar *window4* (tv:make-window 'scroller2
                                  ':x 700
                                  ':y 50
                                  ':width 359
                                  ':height 350
                                  ':default-character-style '(:dutch
:bold :small)
                                  ':item-type-alist alist-alpha
                                  ':end-of-page-mode :truncate))


COMMENT:   This definition contains emergency response operator
guidelines for many events

(setf (aref kb 4)
      '((start)
      (sgtr
        ((procedure
           ((1 "1. Identify Ruptured Steam Generator")
            (2 "2. Isolate Ruptured Steam Generator"
              ("2.1 WHEN in the narrow range, THEN stop all AFW flow
to ruptured steam generators" (confirm))
              ("2.2 Close ruptured steam generator main steamline
isolation valve and bypass valve" (confirm) "Close non-ruptured steam
generator main steamline isolation valves and bypass valves.  Use
non-ruptured steam generator PORVs for steam dump." (confirm))
              ("2.3 Verify ruptured steam generator PORVs closed"
(rule (PORV.status steam.gen.1 closed 1))
              "Manually close ruptured steam generator PORV" (confirm))
              ("2.4 Close ruptured steam generator steam supply valve
to turbine-driven AFW pump" (confirm)))
            (3 "3. Check Pressurizer PORV Block Valves:"
```

```
                ("3.1 Power available to block valves" (confirm)
"Restore power to block valves" (confirm))
                ("3.2 OPEN block valves" (confirm) "Manually close
PORVs.  If any valve cannot be closed, Then manually close its block
valve" (confirm)))
            (4 "4. Check Pressurizer PORVs:"
             ("4.1 Close PORVs" (rule (PORV.status pressurizer closed
1)) "Manually close PORVs. IF any valve cannot be closed, then
manually close its block valve." (confirm)))
             (c1 "CAUTION: IF any pressurizer PORV opens because of
high RCS pressure,  repeat step 4 after pressure drops below PORV
setpoint. Seal injection flow should be maintained to all RCPs.")
            (5 "5. Check if RCP Should Be Stopped:"
             ("5.1 If SI running - CHECK FOR FLOW OR PUMP BREAKER
INDICATOR LIGHTS LIT")
              ("5.1.1 Charging/SI" (confirm) "DO NOT STOP RCPs." (6))
              ("5.1.2 High-head/SI" (confirm) "DO NOT STOP RCPs." (6))
               ("5.2 RCS pressure - EQUAL TO OR LESS THAN ??? PSIG"
(confirm) "DO NOT STOP RCPs." (6))
              ("5.3 Stop all RCPs" (confirm)))
            (6 "6 Check If Low-head SI Pumps Should Be Stopped:"
             ("6.1 Check RCS pressure")
              ("6.1.1 Pressure - GREATER THAN 1000 PSIG" (rule ((rule
(pressure reactor.vessel > number 1000)) 1))
             "LOSS OF COOLANT ACCIDENT" (LOCA 1))
               ("6.1.2 Pressure - STABLE OR INCREASING" (rule ((or
(pressure.gradient  reactor.vessel  normal)  (pressure.gradient
reactor.vessel increasing))) 1) (7))
              ("6.2 Reset SI" (confirm))
                ("6.3 Stop low-head SI pumps amd place in standby"
(confirm)))
            (7 "7. Check Electrical Power and Air Supply Available
To Essential Equipment"
              ("7.1 Establish power supplies as necessary" (confirm)))
             (8 "8. Check Secondary System Integrity:"
               ("8.1 RCS hot leg temperature - GREATER THAN 500 F"
(rule ((rule (temperature hot.leg1 > number 500)) 1))
             "RCS  hot  leg  temperature  decreasing"  (rule
(temperature.gradient hot.leg1 decreasing 1))
             "Close all main steamline isolation valves and bypass
valves." (confirm)
             "Steam generator pressure continuing to decrease"
(confirm) "SGTR with SECONDARY DEPRESSURIZATION" (SGTR-SD 1))
               ("8.2 ALL steam generator pressures - GREATER THAN ???
PSIG" (confirm)
             "Close all main steamline isolation valves and bypass
valves." (confirm)
             "Steam generator pressure continuing to decrease"
(confirm) "SGTR with SECONDARY DEPRESSURIZATION" (SGTR-SD 1)))
              (c2 "CAUTION:  Alternate water sources for AFW pumps will
benecessary, if CST level is low")
              (9 "9. Check Steam Generator Levels:"
               ("9.1 Narrow range level - GREATER THAN ??? %" (confirm)
"Maintain full AFW flow until narrow range level is greater than ???
%" (confirm))
```

```
            ("9.2 Throttle AFW flow to maintain narrow range level
at ??? %" (confirm)))
             (c3 "CAUTION: DO NOT PROCEED to step 10 until ruptured
steam generator has been identified and isolated.")
             (10 "10. Cooldown Non-ruptured Steam Generators 50 F
Below Ruptured Steam Generator:"
              ("10.1 Determine required non-ruptured steam generator
pressure" (confirm))
              ("10.2 Rapidly dump steam to condenser from non-ruptured
steam generators:" (confirm)
             "Rapidly dump steam with non-ruptured steam generator
PORVs" (confirm))
              ("10.3 Check ruptured steam generator pressure - STABLE
or INCREASING" (confirm)
             "SGTR with SECONDARY DEPRESSURIZATION" (SGTR-SD 1)))
             (c4 "CAUTION: If containment conditions are abnormal, go
to E-1 LOSS OF REACTOR COOLANT, STEP 9.")
             (11 "11. Check RCS Pressure:"
               ("11.1 RCS pressure - AT LEAST 200 PSI GREATER THAN
RUPTURED STEAM GENERATOR PRESSURE" (confirm) "SGTR CONTINGENCIES"
(SGTR-C 1)))
             (12 "12. Depressurize RCS Using Normal Spray:"
                ("12.1 Verify normal spray - AVAILABLE" (confirm)
"Failure.." (14))
                ("12.2 Open normal spray valves" (confirm) "Failure.."
(14))
                  ("12.3 Verify RCS pressure - DECREASING" (rule
(pressure.gradient reactor.vessel decreasing 1))
             "Close spray valves" (14)))
               (13 "13. Check If RCS Depressurization Should Be
Stopped:"
              ("13.1 RCS pressure - LESS THAN OR EQUAL TO RUPTURED
STEAM GENERATOR PRESSURE
-OR-
Pressurizer level - GREATER THAN ??? %" (confirm))
                ("13.2 Stop RCS depressurization by closing spray
valves" (confirm))
                ("13.3 Check pressurizer level - GREATER THAN ??? %"
(rule ((rule (level pressurizer > number ???)) 1))
             "SGTR CONTINGENCIES" (SGTR-C 1))
                     ("13.4 Verify RCS pressure - INCREASING"
(pressure.gradient reactor.vessel increasing)
             "Stop RCPs in loops with spray line connections"
(confirm)))
             (14 "14. Depressurize RCS Using One Pressurizer PORV"
               ("14.1 Open one pressurizer PORV" (confirm) "If RCS
cannot be depressurized using any PORV, THEN use auxiliary spray."
(confirm)))
              (15 "15. Check Of RCS Depressurization Should Be
Stopped:"
              ("15.1 RCS pressure - LESS THAN OR EQUAL TO RUPTURED
STEAM GENERATOR PRESSURE
-OR-
Pressurizer level - GREATER THAN ??? %" (confirm))
              ("15.2 Stop RCS depressurization:")
```

```
            ("15.2.1 Close PORV" (PORV.status pressurizer closed)
"Close block valve" (confirm))
            ("15.2.2 Close auxiliary spray valve" (confirm) "Isolate
auxiliary spray line" (confirm))
            ("15.3 Check pressurizer level - GREATER THAN ??? %"
(rule ((rule (level pressurizer > number ???)) 1))
          "SGTR CONTINGENCIES" (SGTR-C 1))
                  ("15.4 Verify RCS pressure - INCREASING"
(pressure.gradient reactor.vessel increasing)
          "Check PRT conditions for RCS leak" (confirm) "LOSS OF
COOLANT" (LOCA 1)))
            (c5 "CAUTION: If PRT integrity is lost, abnormal
containment
conditions may not be reliable indications of a loss of reactor
coolant.")
          (16 "16. Check If SI Can Be Terminated:"
           ("16.1 RCS pressure - INCREASE BY 200 PSI" (confirm) "DO
NOT TERMINATE SI. Check if pressurizer level is increasing" (confirm)
"SGTR CONTINGENCIES" (SGTR-C 1))
            ("16.2 Pressurizer level - GREATER THAN 30 %" (rule
((rule (level pressurizer > number 30))))
          "DO NOT TERMINATE SI. SGTR CONTINGENCIES" (SGTR-C 1))
            ("16.3 RCS subcooling - GREATER THAN ??? F" (confirm)
"DO NOT TERMINATE SI.") (16))
            (c6 "CAUTION: Do not proceed to step 17 until all
conditions in step 16 are met.")
          (17 "17. Terminate SI:"
            ("17.1 Go to ES-3.1, SI TERMINATION FOLLOWING STEAM
GENERATOR TUBE RUPTURE" (SI-T-SGTR 1)))
          (18 "18. Check If Condensor Can Be Used"
            ("18.1 Condensor - AVAILABLE" (confirm) "Attempt to
restore condensor" (confirm)
             "Evaluate adequacy of radiation hazard" (confirm)
"Alternate Cooldown Procedures required" (SGTR-ACP 1)))
          (19 "19. Shutdown Margin"
            ("19.1 Verify Adequate Shutdown Margin" (confirm)
"Borate as necessary" (confirm)))
            (c7 "CAUTION: Steps 20 through 23 must be performed
simula-
taneously to avoid loss of pressurizer level control")
          (20 "20. Initiate RCS Cooldown to 350 F"
            ("20.1 Maintain cooldown rate - LESS THAN 50 F/HR"
(confirm))
            ("20.2 Dump steam from non-ruptured steam generators
to condenser" (confirm)
             "Dump steam with non-ruptured steam-generator PORVs"
(confirm)))
            (c8 "CAUTION: Charging and letdown flows should be
compared
to determine if leakage between the RCS and ruptured steam generator
is stopped.")
          (21 "21. Maintain Pressurizer Level in Normal Operating
Range:"
            ("21.1 Operate chargine and letdown, as necessary"
(confirm)))
```

```
            (22 "22. Depressurize Ruptured Steam Generator"
               ("22.1 Slowly release steam to condenser from ruptured
steam generator" (confirm)
                "Slowly release steam to atmosphere with ruptured
steam generators PORV" (confirm)))
               (c9 "CAUTION: Maintain RCS pressure and temperature
within normal cooldown limits
        IF RCS pressure or pressurizer level drop in an uncontrolled
manner, THEN reinitiate SI and return to step 10.")
            (23 "23. Depressurize RCS:"
               ("23.1 Reduce RCS pressure to maintain RCS/ruptured
steam generator pressures equal")
               ("23.1.1 Use normal pressurizer spray" (confirm) "If
Letdown in service, then use auxiliary spray, if not, then use one
pressurizer PORV." (confirm)))
               (24 "24. Determine If SI Accumulators Should Be
Isolated:"
               ("24.1 RCS pressure - LESS THAN OR EQUAL TO ??? PSIG"
                (rule ((rule (pressure reactor.vessel < number 400))
1)) (20))
                ("24.2 Close all SI accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (25 "25. Check If RHR System Can Be Placed In Service:"
               ("25.1 RCS hot leg temperatures - LESS THAN 350 F IN
NON-RUPTURED LOOPS" (confirm) (20))
                ("25.2 RCS Pressure - APPROXIMATELY 400 PSIG"
(confirm) (21)))
               (c10 "CAUTION: Do not collapse the pressurizer bubble")
               (26 "26. Place RHR System In Service"
                 ((confirm)))
               (27 "27. Continue Cooldown To Cold Shutdown:"
                  ("27.1 Cooldown using RHR"  (confirm))
                  ("27.2 At least on RCP - RUNNING" (confirm) "Continue
dumping steam from non-ruptured steam generators until they have
stopped steaming." (confirm)))
            (28 "28. Check RCS Temperature:"
               ("28.1 Temperature - LESS THAN ??? F" (rule ((rule
(temperature reactor.vessel < number 500)) 1)) (27))
                ("28.2 Stop all RCPs" (confirm))
                ("28.3 Cooldown pressurize")
                 ("28.3.1 Spray pressurizer with auxiliary Spray"
(confirm)))
            (29 "29. Maintain Cold Shutdown Conditions")))))
      (si-t-sgtr
        ((procedure
           ((1 "1. Reset SI")
            (2 "2. Stop SI Pumps And Place In Standby:"
             ("2.1 Stop Low-head SI pumps" (confirm))
             ("2.2 Stop High-head SI pumps" (confirm))
             ("2.3 All but one charging/SI pump" (confirm)))
            (3 "3. Establish Charging/SI Pump Miniflow:"
             ("3.1 Verify CCW flow to seal water heat exchanger"
(confirm))
              ("3.2 Open miniflow isolation valves" (confirm)))
            (4 "4. Isolate BIT:"
```

```
            ("4.1 Close inlet isolation valves" (confirm))
            ("4.2 Close outlet isolation valves" (confirm)))
           (5 "5. Verify SI Reinitiation NOT Required:"
            ("5.1 RCS subcooling GREATER THAN ??? F" (confirm)
"Manually operate SI pumps, as required for maintaining subcooling"
(confirm) "STEAM GENERATOR TUBE RUPTURE" (sgtr 10))
            ("5.2 Pressurizer level GREATER THAN 20%" (rule ((rule
(level pressurizer > number 20)) 1))
             "Manually operate SI pumps, as required for maintaining
pressurizer level" (confirm)
             "Manually reinitiate SI." "STEAM GENERATOR TUBE
RUPTURE" (sgtr 10)))
           (6 "6. Verify Offsite Power Available"
            ("6.1 Offsite power available" (rule ((rule (power
generator > number 5)) 1)) "Try to restor offsite power"
             (confirm) "Manually load following equipment on the
diesel generators:"))
           (7 "7. Reset Containment Isolation Phase A")
           (8 "8. Establish Charging:"
            ("8.1 Close charging flow control valve" (confirm))
            ("8.2 Open charging line isolation valves" (confirm))
            ("8.3 Open charging flow control valve to establish
desired flow" (confirm)))
           (9 "9. Establish Letdown:"
            ("9.1 Open letdown line containment isolation valves"
(confirm))
            ("9.2 Open letdown line isolation valves" (confirm))
            ("9.3 Open letdown orifice isolation valves, as
appropriate"))
           (10 "10. Align Charging/SI Pump Suction to VCT:"
             ("10.1 Open VCT outlet isolation valves" (confirm))
             ("10.2 Close RWST outlet isolation valves" (confirm)))
           (11 "11. Check VCT Makeup Control System:"
             ("11.1 Makeup set for automatic control" (confirm)
"Adjust controls, as appropraite." (confirm))
            ("11.2 Makeup for GREATER THAN RCS boron concentration"
(confirm) "Adjust controls, as appropriate"))
           (12 "12. Check RCP Cooling:"
             ("12.1 RCP seal injection flow - NORMAL" (confirm)
"Adjust charging line hand control valve, as necessary."
             (confirm))
             ("12.2 RCP CCW system flow - NORMAL" (confirm)
"Establish CCW flow to RCPs:
1) Reset containment isolation Phase B, if necessary.
2) Open approriate CCW system isolation valves"))
           (13 "13. Check Non-Faulted Steam Generator Levels:"
            ("13.1 Narrow range level - GREATER THAN ???" (confirm)
"Maintain full AFW flow until narrow range level is greater than ???"
(confirm))
            ("13.2 Throttle AFW flow to maintain narrow range at
???" (confirm)))
           (14 "14. Check CST Level"
            ("14.1 CST level GREATER THAN ??? %" (confirm) "Switch
to alternate AFW water supply."))
           (15 "15. Establish Pressurizer Pressure Control:"
```

```
            ("15.1 Energize pressurizer heaters, as necessary to
maintain pressure" (confirm)))
          (16 "16. Check RCP Status:"
            ("16.1 At least on RCP - Running" (confirm) "Attempt to
restore on RCP" (confirm) "Verify natural circulation:
a) RCS subcooling GREATER THAN ??? F
b) Steam pressure stable
c) RCS hot leg temperature STABLE or SLOWLY DECREASING
d) Core exit TCs - STABLE or SLOWLY DECREASING
e) RCS cold leg temperature - NEAR SATURATION TEMPERATURE FOR STEAM
PRESSURE" (confirm)
             "Increase dumping steam" (confirm)))
          (17 "17. Check Intermediate Range Flux:"
            ("17.1 Flux BELOW ???" (confirm) "Shutdown unnecessary
plant equipment" (confirm))
             ("17.2 Verify source range detectors re-energized"
(confirm) "Manually reenergize source range detectors."
              (confirm))
             ("17.3 Transfer nuclear recorders to source range
scale" (confirm)))
          (18 "18. Shutdown Unnecessary Plant Equipment")
          (19 "19. Verify SI Reinitiation NOT Required:"
            ("19.1 RCS subcooling GREATER THAN ??? F" (confirm)
"Manually operate SI pumps, as required to
maintain subcooling" (confirm) "STEAM GENERATOR TUBE RUPTURE" (sgtr
10))
             ("19.2 Pressurizer level GREATER THAN 20%" (rule ((rule
(level pressurizer > number 20)) 1))
               "Manually operate SI pumps, as required for
maintaining pressurizer level" (confirm)
               "Manually reinitiate SI." "STEAM GENERATOR TUBE
RUPTURE" (sgtr 10)))
          (20 "20. Continue with Procedure In Effect"
            ((return)))))))
    (sgtr-sd
      ((procedure
         ((1 "1. Verify Main Steamline Isolation"
           ("1.1 Main steamline isolation valves CLOSED" (confirm)
"Manually close valves")
            ("1.2 Main steamline bypass valves CLOSED" (confirm)
"Manually close valves"))
          (2 "2. Identify Faulted Steam Generator"
           ("2.1 a) Steam generator pressure LESS THAN ???
   b) Steam generator pressure DECREASING
   c) Inspect steam generators and main steamline" (confirm) "STEAM
GENERATOR TUBE RUPTURE" (sgtr 9)))
          (3 "3. Try to Stop Steam Release From Faulted Steam
Generator"
           ("3.1 Steam generator PORVs CLOSED" (confirm) "Manually
close steam generator PORVs" (confirm)
            "Try isolating that PORV while continuing mitigation
procedures" (confirm)))
          (4 "4. Check Faulted Steam Generator Pressure:"
           ("4.1 Any faulted steam generator pressure STABLE or
DECREASING" (confirm) "STEAM GENERATOR TUBE RUPTURE"
```

```
            (sgtr 10)))
          (5 "5. Stop AFW Flow To Faulted Steam Generators:"
           ("5.1 Follow plant specific steps" (confirm)))
          (c1 "Alternate water sources for AFW will be necessary if
CST level is low")
          (6 "6. Check Non-Faulted Steam Generator Level"
           ("6.1 Narrow range level GREATER THAN ??? %" (confirm)
"Maintain full AFW flow until narrow range level is greater than ???
%" (confirm))
           ("6.2 Throttle AFW to maintain narrow range level at ??
%" (confirm)))
          (7 "7. Identify Ruptured Steam Generator"
           ("7.1 a) Unexpected rise in steam generator narrow range
level
    b) High radiation from any steam generator blowdown lines
    c) High radiation from any steam generator sample
    d) High radiation from any steam generator steamline" (confirm)))
          (8 "8. Isolate Ruptured Steam Generator"
           ("8.1 When in narrow range, stop all AFW flow to
ruptured generator" (confirm))
           ("8.2 Verify ruptured steam generator PORV CLOSED"
(confirm) "Manually close ruptured steam generator PORV"
            (confirm))
           ("8.3 Close ruptured steam generator steam supply valve
to
turbine steam supply valve to turbine-drive AFW pump" (confirm)))
          (9 "9. Check Intact Steam Generator Levels:"
           ("9.1 Narrow range level GREATER THAN ??? %" (confirm)
"Maintain full AFW flow until narrow range level is greater than ???
%" (confirm))
           ("9.2 Throttle AFW to maintain narrow range level at ??
%" (confirm)))
          (10 "10. Verify Ruptured Steam Generaor is Faulted"
            ((confirm) "STEAM GENERATOR TUBE RUPTURE" (sgtr 10)))
          (11 "11. Check RCS Pressure:"
           ("11.1 RCS pressure - GREATER THAN SI ACCUMULATOR
PRESSURE" (confirm) "SGTR with CONTINGENCIES" (sgtr-c 1)))
          (c2 "CAUTION: If main steamline isolation valve or bypass
valve for any ruptured steam generator fails open, the main steamline
isolation valves and bypass valves for non-ruptured steam generators
must remain closed.")
           (12 "12. Depressurized Non-Ruptured Steam Generators to
250 psig:"
            ("12.1 Rapidly dump steam to condenser form non-
ruptured steam generators" (confirm)
             "Rapidly dump steam with non-ruptured steam generator
PORVs" (confirm)))
           (c3 "CAUTION: If containment conditions are abnormal,
LOCA"
            ("Verify normal containment conditions" (confirm)
"LOCA" (loca 1)))
          (13 "13. Depressurize RCS Using Normal Spray:"
           ("13.1 Verify normal spray AVAILABLE" (confirm) (15))
           ("13.2 Open normal spray valves" (confirm) (15))
```

```
                ("13.3 Verify RCS pressure DECREASING"
(pressure.gradient reactor.vessel decreasing) "Close spray valves"
(15)))
                (14 "14. Check if RCS Depressurization Should Be
Stopped:"
              ("14.1 RCS subcooling LESS THAN OR EQUAL TO 50 F
-OR-
Pressurizer level GREATER THAN ??? %" (confirm) "Continue
depressurization until condition is met")
                ("14.2 Stop RCS depressurization by closing spray
valves" (confirm))
                ("14.3 Check pressurizer level GREATER THAN ??? %"
(confirm) "SGTR with CONTINGENCIES" (sgtr-c 1))
                 ("14.4 Verify RCS pressure INCREASING"
(pressure.gradient reactor.vessel increasing)
                  "Stop RCPs in loops with spray line connections"
(confirm))
                 ((17)))
              (15 "15. Depressurize RCS Using One Pressurizer PORV:"
                ("15.1 Open one pressurizer PORV" (confirm) "Manually
open PORV" (confirm))
                ("15.2 Verify RCS depressurization" (confirm) "Use
auxiliary spray" (confirm)))
                 (16 "16. Check If RCS Depressurization Should Be
Stopped:"
                ("16.1 RCS subcooling LESS THAN OR EQUAL TO 50 F
-OR-
Pressurizer level GREATER THAN ??? %" (confirm) "Continue
depressurization until either condition met." (confirm))
                ("16.2 Stop RCS depressurization:")
                ("16.2.1 Close PORV" (confirm) "Close PORV block valve"
(confirm))
                  ("16.2.2 Close auxiliary spray valve" (confirm)
"Isolate auxiliary spray line" (confirm))
                  ("16.3 Check pressurizer level GREATER THAN ??? %"
(confirm) "SGTR with CONTINGENCIES" (sgtr-c 1))
                   ("16.4 Verify RCS pressure INCREASING"
(pressure.gradient reactor.vessel increasing)
                    "Check PRT condition for RCS integrity" (confirm)
"LOCA" (loca 9)))
              (17 "17. Check If SI Can Be Terminated:"
                ("17.1 RCS pressure INCREASES by 200 PSI" (confirm) "DO
NOT TERMINATE SI. Confirm pressurizer level INCRESING" (confirm)
"SGTR with CONTINGENCIES" (sgtr-c 1))
                ("17.2 Pressurizer level GREATER THAN ?? %" (confirm)
"DO NOT TERMINATE SI"
                  "SGTR with CONTINGENCIES" (sgrt-c 1))
                ("17.3 RCS subcooling GREATER THAN ?? %" (confirm)))
              (c4 "CAUTION:
1) Do not proceed until all conditions in step 17 are met.
2) Following SI reset, automatic reinition of SI will not occur until
reactor
trip breakers are reset.
3) If offsite power is lost after SI reset, manual action may be
required to restart safeguards equipment." (confirm))
```

```
            (18 "18. Reset SI"
              ((confirm)))
            (19 "19. Reset Containment Isolation Phase A."
              ((confirm)))
            (20 "20. Stop SI Pumps and Place In Standby:"
              ("20.1 Stop low-head SI pumps" (confirm))
              ("20.2 Stop high-head SI pumps" (confirm))
              ("20.3 Stop all but one charging/SI pump" (confirm)))
            (21 "21. Establish Charging/SI Pump Miniflow:"
              ("21.1 Verify CCW flow to seal water heat exchanger"
(confirm))
              ("21.2 Open miniflow isolation valves" (confirm)))
            (22 "22. Isolate BIT:"
              ("22.1 Close inlet isolation valves" (confirm))
              ("22.2 Close outlet isolation valves" (confirm)))
            (23 "23. Realign Charging Flow Path:"
              ("23.1 Close charging line hand" (confirm))
              ("23.2 Open charging line isolation valves" (confirm)))
            (24 "24. Check If Charging Flow Should Be Established:"
              ("24.1 RCS subcooling LESS THAN or EQUAL to 50 F
          -OR-
Pressurizer Level LESS THAN or EQUAL TO 30 %" (confirm))
              ("24.2 Control charging flow to maintain pressurizer
level approximately constant" (confirm)
               "Manually operate SI pumps, as necessary" (confirm)))
            (25 "25. Verify SI Reinitiation Not Required:"
              ("25.1 RCS subcooling GREATER THAN ??? F" (confirm)
"Manually operate SI pumps, as required.  Confirm that subcooling can
be maintained" (confirm) "Manually reinitiate SI" (12))
              ("25.2 Pressurizer level GREATER THAN 20%" (confirm)
"Manually operate SI pumps, as required.  Confirm maintainability of
pressurizer level" (confirm) "Manually reinitiate SI" (12)))
            (26 "26. Verify Offsite Power Available:"
              ("26.1 Follow plant specific procedures" (confirm)
"Start Diesel Generators:" (confirm)
               "Bend over and kiss your ass goodbye"))
            (27 "27. Try to Establish Pressurizer Pressure Control:"
              ("27.1 Energize heaters as necessary to maintain
pressure" (confirm)))
            (n1 "NOTE:  RCPs should be operated in order of priority
to provide pressurizer spray." (confirm))
            (28 "28. Check RCP Status:"
              ("28.1 At least one RCP-RUNNING" (confirm) "Manually
start one RCP" (confirm))
              ("28.2 If more than one RCP is running, then stop all
but one" (confirm)))
            (29 "29. Try to Initiate Blowdown From Ruptured Steam
Generator(s):"
              ("29.1 Follow plant specific steps" (confirm)))
            (30 "30. Verify Adequate Shutdown Margin"
              ("30.1 If necessary, Borate" (confirm)))
            (31 "31. Rapidly Cooldown RCS to 350 F:"
              ("31.1 Maintain cooldown rate to LESS THAN 100 F/HR"
(confirm))
```

```
            ("31.2 Dump steam from non-ruptured steam generators to
condenser" (confirm) "Dump steam with non-ruptured steam generator
PORVs" (confirm)))
            (32 "32. Control Charging Flow to Maintain Pressurizer
Level Nearly Constant:"
             ("32.1 Manually operate SI pumps as necessary.  Confirm
maintainability of pressurizer level" (confirm)
              "Reinitiate SI" (17)))
            (33 "33. Check Ruptured Steam Generator(s) Pressure:"
             ("33.1 Any Ruptured steam generator pressure LESS THAN
RCS PRESSURE OR DECREASING" (confirm)
              "If all ruptured steam generator pressures greater
than or equal to RCS pressure and stable"
              (confirm-) "SGTR with CONTINGENCIES" (sgtr 17)))
            (34 "34. Depressurize Ruptured Steam Generator(s)  If
Necessary:"
             ("34.1 Slowly release steam to condenser from ruptured
steam generator(s)" (confirm)
              "Slowly release steam to atmosphere with ruptured
steam generator PORV" (confirm)))
            (35 "35. Isolate SI Accumulators:"
             ("35.1 Close all SI accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (36 "36. Depressurize RCS:"
             ("36.1 Use normal pressurizer spray" (confirm) "Use one
pressurizer PORV" (confirm)
              "Use auxiliary spray" (confirm)))
             (37 "37.  Check if RCS Depressurization Should be
Stopped:"
             ("37.1 RCS subcooling LESS THAN OR EQUAL TO 50 F
               -OR-
RCS PRESSURE LESS THAN OR EQUAL TO ANY RUPTURED STEAM GENERATOR
PRESSURE" (confirm) "Continue depressurization until either condition
is met." (confirm))
             ("37.2 Stop RCS Depressurization" (confirm)))
            (38 "38. Check if RHR System can be Placed in Service:"
             ("38.1 RCS hot leg temperatures LESS THAN 350 F IN NON-
RUPTURED LOOPS" (confirm) (31))
              ("38.2 RCS pressure nearly 400 psig" (confirm) "RCS
pressure less than 400 psig" (confirm) (32)))
            (39 "39. Place RHR System in Service:"
             "39.1 Follow plant specific steps" (confirm))
            (40 "40. Continue Rapid Cooldown to Cold Shutdown:"
             ("40.1 Maintain cooldown rate LESS THAN 100 F/HR"
(confirm))
             ("40.2 Cooldown using RHR system" (confirm))
             ("40.3 Continue dumping steam from non-ruptured steam
generators" (confirm)))
            (41 "41. Check RCP Status:"
             ("41.1 At lease one RCP-RUNNING" (confirm) (42))
             ("41.2 RCS pressure GREATER THAN ??? PSIG" (confirm)
              "Stop depressurization.  Maintain RCS pressure at ???
psig" (45)))
            (42 "42. Depressurize Ruptured Steam Generator(s):"
```

```
            ("42.1 Slowly release steam to condenser from ruptured
steam generator(s)" (confirm)
             "Slowly release steam to atmosphere with ruptured
steam generator(s) PORV" (confirm)))
          (43 "43. Depressurize RCS:"
           ("43.1 Use normal pressurizer spray" (confirm) "Use one
pressurizer PORV" (confirm)
            "Use auxiliary spray" (confirm))
            ("43.2 Turn off pressurizer heaters, as necessary"
(confirm))
           ("43.3 Control charging flow to maintain pressurizer
level approximately constant" (confirm)
            "Manually operate SI pumps, as necessary." (confirm)))
            (44 "44. Check if RCS Depressurization Should be
Stopped:"
           ("44.1 RCS subcooling LESS THAN OR EQUAL TO 50 F
             -OR-
RCS PRESSURE LESS THAN OR EQUAL TO ANY RUPTURED STEAM GENERATOR
PRESSURE" (confirm)
            "Continue depressurization until either condition is
met." (confirm))
           ("44.2 Stop RCS depressurization" (confirm)))
          (45 "45. Check RCS Hot Leg Temperatures:"
           ("45.1 Temperature -LESS THAN 200 F" (rule ((rule
(temperature hot.leg1 < number 200)))) (40))
           ("45.2 Stop all RCPs" (confirm)))
          (46 "46. Depressurize RCS to Stop Break Flow:"
           ("46.1 Depressurize ruptured steam generator(s), as
necessary, by dumping steam to condenser from ruptured steam
generator(s)" (confirm) "Dump steam with ruptured steam generator(s)
PORV" (confirm))
            ("46.2 Cooldown pressurizer with auxiliary spray"
(confirm))
           ("46.3 When pressurizer temperature reaches 200 F, stop
chargine pumps" (confirm)))
          (47 "Maintain Cold Shutdown Conditions")))))
    (sgtr-c
      ((procedure
        ((1 "1. Check Secondary System:"
          ("1.1 All Steam Generator Pressures GREATER THAN ???
PSIG" (confirm) "Close all main steamline isolation valves and bypass
valves." (confirm) "Confirm stabilization of steam generator
pressures" (confirm) (20)))
          (n1 "NOTE: Steps 2-19 provide contingencies for recovery
from a SGTR without an uncontrolled secondary steam release.")
          (2 "2. Establish Pressurizer Level Using Normal Spray"
           ("2.1 Verify normal spray - Available" (confirm) (4))
           ("2.2 Open normal spray valves" (confirm) (4))
           ("2.3 Verify pressurizer level - INCREASING" (confirm)
"Close spray valves" (confirm-) (4)))
          (3 "3. Check if RCS Depressurization Should be Stopped:"
           ("3.1 Pressurizer level - GREATER THAN 40 %" (confirm)
"Continue depressurization" (confirm))
            ("3.2 Stop RCS depressurization by closing spray valves"
(confirm))
```

```
            ("3.3 Pressurizer level - STABLE OR DECREASING"
(confirm) "Stop RCPs in loops with spray line connections."
              (confirm))
             ((6)))
             (4 "Establish Pressurizer Level Using One Presssurizer
PORV:"
              ("4.1 Open one pressurizer PORV" (confirm) "Use
auxiliary spray" (confirm)))
            (5 "5. Check if RCS Depressurization Should be Stopped:"
              ("5.1 Pressurizer level - GREATER THAN 40 %" (confirm)
"Continue depressurization" (confirm))
              ("5.2 Stop RCS depressurization by closing PORV and the
auxiliary spray valve" (confirm)
               "Close PORV block valve and/or Isolate auxiliary spray
line" (confirm))
              ("5.3 Pressurizer level - STABLE OR DECREASING"
(confirm)
               "Check PRT conditions for indication of RCS integrity"
(confirm) (loca 1)))
              (c1 "CAUTION: If PRT integrity is lost, abnormal
containment conditions may not be reliable indications of loss of
reactor coolant")
             (6 "6. Check if SI can be Terminated:"
              ("6.1 Pressurizer level - GREATER THAN 20% AND STABLE"
(confirm) "Do not terminate SI" (confirm))
              ("6.2 RCS subcooling - GREATER THAN ??? F" (confirm) "Do
not terminate SI" (confirm)))
              (c2 "CAUTION: Do not proceed to step 7 until all
conditions in step 6 are met"
              ((confirm) (6)))
             (7 "7. Terminate SI:"
              ("SI TERMINATION" (si-t-sgtr 1)))
             (8 "8. Turn off Pressurizer Heaters"
              ((confirm)))
             (9 "9. Check if Condenser can be Used:"
              ("9.1 Confirm condenser availability" (confirm) "Try to
restore condenser" (confirm)
               "Confirm 10 CFR 20 limits are not exceeded for releases
from ruptured steam generator"
               (confirm) "SGTR ALTERNATE COOLDOWN" (sgtr-acp 1)))
             (10 "10. Verify Adequate Shutdown Margin"
              ("10.1 Borate as necessary" (confirm)))
              (c3 "CAUTION: 1) Steps 11,12 and 13 must be performed
simultaneously to avoid loss of pressurizer level control.
2) Maintain RCS Pressure and temperature within normal cooldown
limits.
3) If RCS pressure or pressurizer level drop in an uncontrolled
manner, THEN reinitiate SI and return to step 1")
             (11 "Initiate Rapid RCS Cooldown To 350 F:"
              ("11.1 Maintain cooldown rate - LESS THAN 100 F/HR"
(confirm))
              ("11.2 Dump stam from non-ruptured steam generators to
condenser" (confirm)
               "Dump steam from non-ruptured steam generator PORVs."
(confirm)))
```

```
            (c4 "CAUTION: Charging and let down flows should be
compared to determine if leakage between
the RCS and the Ruptured Steam Generator is stopped.")
            (12 "12. Depressurize RCS to Maintain Pressurizer Level:"
              ("12.1 Reduce RCS pressure to maintain pressurizer
level between 40% and 60% using normal pressurizer spray"
                (confirm) "If letdown is in service use auxiliary
spray, else use pressurizer PORV" (confirm)))
            (13 "13. Depressurizer Reptured Steam Generator(s):"
              ("13.1 Slowly release steam to condenser from ruptured
steam generator(s)" (confirm)
               "Slowly release steam to atmospher with ruptured steam
generator(s) PORV" (confirm)))
              (14 "14. Determine if SI Accumulators Should be
Isolated:"
               ("14.1 RCS pressure LESS THAN OR EQUAL TO ??? PSIG"
(confirm) (11))
               ("14.2 Close all SI accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (15 "15. Check if RHR System can be Placed in Service:"
              ("15.1 RCS hot leg temperature - LESS THAN 350 F IN
NON-RUPTURED LOOPS" (confirm) (11))
              ("15.2 RCS pressure - APPROXIMATELY 400 PSIG" (confirm)
"Confirm that pressure is less than 400 psig"
                (confirm) (12)))
            (c5 "CAUTION: Do not collapse the pressurizer bubble"
              ((confirm)))
            (16 "16. Place RHR System in Service:"
              ("16.1 Follow plant specific guidelines" (confirm)))
            (17 "17. Continue Cooldown to Cold Shutdown:"
              ("17.1 Cooldown using RHR" (confirm))
              ("17.2 At least one RCP RUNNING" (confirm) "Continue
dumping steam from non-ruptured steam generators until they have
stopped steaming" (confirm)))
            (18 "18. Check RCS Temperatures:"
              ("18.1 RCS Temperature - LESS THAN ??? F" (confirm)
(17))
              ("18.2 Stop all RCPs" (confirm))
              ("18.3 Cooldown pressurizer by spraying pressurizer
with auxiliary spray" (confirm))
              ("18.4 When pressurizer temperture reaches 200 F, stop
chargine pumps" (confirm)))
            (n2 "19. NOTE: Steps 20-30 provide contingencies for
recovery from a SGTR with an uncontrolled secondary steam release"
              ((31)))
              (20 "20. Cooldown RCS by Depressurizing ALL Steam
Generators To 140 PSIG:"
              ("20.1 Rapidly dump steam to condenser (follow plant
specific procedures)" (confirm)
                "Rapidly dump steam with steam generator PORVs"
(confirm)))
                (c6 "CAUTION: Steps 21-29 must be performed
simultaneously to avoid loss of pressurizer level control"
              ("Confirm adequate containment conditions" (confirm)
(loca 9)))
```

```
            (21 "21. Depressurize RCS To Establish Pressurizer
Level:"
              ("21.1 Open one pressurizer PORV" (confirm) "Use
auxiliary spray" (confirm))
              ("21.2 When pressurizer level is greater than 40%, stop
RCS depressurization" (confirm))
              ("21.3 Verify PORV CLOSED" (confirm) "Close PORV block
valve" (confirm)))
            (22 "22. Verify Adequate Shutdown Margin"
              ("22.1 Borate as necessary" (confirm)))
            (23 "23. Continue Cooldown to Cold Shutdown"
              ("23.1 Maintain cooldown rate - LESS THAN 100 F/HR"
(confirm))
              ("23.2 Dump steam to condenser (follow plant specific
procedures)" (confirm)
               "Dump steam with steam generator PORVs" (confirm))
              ("23.3 Cooldown using RHR system if in service"
(confirm)))
            (24 "24. Check if RHR System can be Placed in Service:"
              ("24.1 RCS hot leg temperatures - LESS THAN 350 F"
(rule ((rule (temperature hot.leg1 < number 350)) 1)) (25))
              ("24.2 RCS pressure - LESS THAN 400 PSIG" (rule ((rule
(pressure reactor.vessel < number 400)) 1)) (25))
              ("24.3 Place RHR system in service per plant specific
procedure" (confirm)))
            (25 "25. Check if SI Accumulators Should be Isolated:"
              ("25.1 RCS pressure - EQUAL TO OR LESS THAN 200 PSIG"
               (rule ((rule (pressure reactor.vessel <= number 200))
1)) (26))
               ("25.2 Close all SI accumulator isolation valves"
(confirm)
               "Vent any unisolated accumulators before proceedings"
(confirm)))
            (26 "26. Check RCS Hot Leg Temperatures:"
              ("26.1 Temperatures - GREATER THAN 200 F" (rule ((rule
(temperature hot.leg1 > number 200)) 1)) (30)))
            (27 "27. Check RCS Conditions:"
              ("27.1 Pressurizer level GREATER THAN 20%" (rule ((rule
(level pressurizer > number 20)) 1)) (21))
               ("27.2 Pressurizer level STABLE OR INCREASING"
(confirm) (21))
               ("27.3 RCS subcooling GREATER THAN 50 F" (confirm)
(21)))
            (28 "28. Isolate all SI accumulators"
              ("28.1 Close all SI accumulators" (confirm) "Vent any
unisolated accumulators" (confirm)))
            (29 "29. Check Pressurizer Level:"
               ("29.1 Pressurizer level STABLE OR INCREASING"
(confirm) (21))
              ("29.2 SGTR With Secondary Depressurization, Step 18"
(sgtr-sd 18)))
               (30 "Depressurize RCS To Ruptured Steam Generator
Pressure:"
              ("30.1 Verify all SI accumulators isolated" (confirm)
"Close all SI accumulator isolation valves" (confirm)
```

```
                "Vent any unisolated accumulator before proceeding"
(confirm))
                ("30.2 Cooldown pressurizer with auxiliary spray"
(confirm))
               ("30.3 When pressurizer temperature reaches 200 F, Stop
all SI pumps" (confirm)))
              (31. "31. Maintain Cold Shutdown Conditions")))))
       (sgtr-acp
         ((procedure
            ((c1 "CAUTION: If pressurizer is water solid, stable plant
conditions must be maintained while heating pressurizer")
             (n1 "NOTE: Have foldout page out")
             (1 "1. Check Pressurizer Water Temperature:"
              ("1.1 Water Temperature EQUAL TO SATURATION TEMPERATURE
OF RUPTURED STEAM GENERATOR" (confirm)
               "Establish required pressurizer water temperature
before continuing" (confirm)))
             (2 "2. Check Pressurizer Level:"
              ("2.1 Level - GREATER THAN OR EQUAL TO 35%" (rule ((rule
(level pressurizer >= number 35)) 1))
               "Increase charging flow until 35% level is reached"
(confirm)))
             (3 "3. Equalize Charging and Letdown Flows:"
              ("3.1 Take manual control of charging and letdown"
(confirm))
               ("3.2 Adjust total charging pump flow EQUAL to letdown
and seal leakoff flows" (confirm)))
               (n2 "NOTE: Flow balance between chargin and letdown
should be maintained throughout this guideline")
             (4 "4. Initiate RCS Cooldown:"
              ("4.1 Maintain cooldown rate - LESS THAN 50 F/HR"
(confirm))
               ("4.2 Maintain RCS pressure - EQUAL TO RUPTURED STEAM
GENERATOR PRESSURE" (confirm))
               ("4.3 Dump steam to condenser from non-ruptured steam
generators (follow plant specific procedures"
               (confirm) "Dump steam with non-ruptured steam generator
PORV" (confirm)))
             (5 "5. Check Pressurizer Level:"
              ("5.1 Level - LESS THAN OR EQUAL TO 25%" (rule ((rule
(level pressurizer <= number 25)) 1))
               "Continue cooldown until level reaches 25%" (confirm)))
             (6 "6. Stop RCS Cooldown"
              ((confirm)))
             (c2 "CAUTION: 50 F subcooling must be maintained at all
times")
              (7 "7. Backfill RCS From Ruptured Steam Generator as
Follows:"
               ("7.1 Check ruptured steam generator narrow range level
- GREATER THAN 25%" (confirm)
                "Refill ruptured steam generator to 65%" (confirm))
               ("7.2 Reduce RCS pressure approximately 50 psi less than
ruptured steam generator pressure by using normal pressurizar spray"
(confirm) "Use auxiliary spray" (confirm) "Use one pressurizer PORV"
(confirm)))
```

```
            (8 "8. Check Pressurizer Level:"
              ("8.1 Level - GREATER THAN OR EQUAL TO 70%" (confirm)
(7)))
            (9 "9. Check RCS Pressure"
              ("9.1 RCS pressure EQUAL TO RUPTURED STEAM GENERATOR
PRESSURE" (confirm)
                "Increase RCS pressure to ruptured steam generator
pressure" (confirm)))
            (10 "10. Determine if SI Accumulators Should be Isolated"
              ("10.1 RCS pressure - LESS THAN OR EQUAL TO ??? PSIG"
(confirm) (11))
                ("10.2 Close all SI accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (11 "11. Check Charging and Letdown Flows:"
              ("11.1 Total charging pump flow - EQUAL TO LETDOWN AND
SEAL LEAKOFF FLOWS" (confirm)
                "Adjust flow as necessary" (confirm)))
            (12 "12. Verify Adequate Shutdown Margin"
              ("12.1 Borate as necessary" (confirm)))
            (13 "Check RCS/Ruptured Steam Generator Pressure:"
              ("13.1 Pressure LESS THAN OR APPROXIMATELY 400 PSIG"
(confirm) "RCS Temperature GREATER THAN 350 F"
                (confirm '(4)) "1)Continue cooldown per steps 4 to 13
until steam generators stop steaming.
2) Maintain stable plant conditions until ruptured steam generator
pressure decays to 400 psig"))
            (14 "14. Check RCS Temperature:"
              ("14.1 Temperature - LESS THAN 350 F" (confirm) "1)
Cooldown RCS below 350 F
2) Maintain pressurizer level with increased charging flow until RCS
is cooled below 350 F" (confirm)))
            (15 "15. Place RHR in Service"
              ("15.1 Follow plant specific procedures" (confirm)))
            (16 "16. Continue RCS Cooldown Using RHR System.")
            (17 "17. Check Pressurizer Level:"
              ("17.1 Level - LESS THAN OR EQUAL TO 25%" (confirm)
                "Continue cooldown until level reaches 25% before
continuing" (confirm)))
            (18 "18. Stop RCS Cooldown")
             (c3 "CAUTION: 50F subcooling must be maintained at all
times")
             (19 "19. Backfill RCS From Ruptured Steam Generator as
Follows:"
              ("19.1 Check ruptured steam generator narrow range
level - GREATER THAN 25%" (confirm)
                "Refill ruptured steam generator to 65%" (confirm))
              ("19.2 Reduce RCS pressure approximately 50 psi less
than ruptured steam generator
pressure by using normal pressurizer spray" (confirm) "Use auxiliary
spray" (confirm) "Use one pressurizer PORV" (confirm)))
            (20 "20. Check Pressurizer Level:"
              ("20.1 Level GREATER THAN OR EQUAL TO 70%" (confirm)
(19)))
            (21 "21. Check RCS Pressure:"
```

```
            ("21.1 RCS pressure - EQUAL TO RUPTURED STEAM GENERATOR
PRESSURE" (confirm)
             "Increase RCS pressure to ruptured generator pressure"
(confirm)))
           (22 "22. Check Charging and Letdown Flows:"
            ("22.1 Total charging pump flow - EQUAL TO LETDOWN AND
SEAL LEAKOFF FLOWS" (confirm)
             "Adjust flows as appropiate" (confirm)))
           (23 "23. Verify Adequate Shutdown Margin"
            ("23.1 Borate as necessary" (confirm)))
           (24 "24. Check RCS Temperature:"
            ("24.1 Temperature - LESS THAN 200 F" (rule ((rule
(temperature reactor.vessel < number 200)) 1)) (16)))
           (25 "25. Maintain Cold Shutdown Conditions")))))
      (loca
       ((procedure
         ((1 "1. Check Pressurizer Level"
           ("1.1 Level - STABLE OR INCREASING" (confirm) "Manually
initiate HPIS" (confirm)))
          (2 "2. Check Sump Pump Status"
           ("2.1 Sump Pump - RUNNING" (confirm) "Nothing's
Wrong"))))))
      (reactor.trip
       ((procedure
         ((1 "1. Check Neutron Density"
           ("1.1 Level - ZERO" (confirm) "Manually scram reactor"))
          (2 "2. Confirm Adequate Cooling"
           ("2.1 Available cooling" (confirm) "Run for your
life"))))))
      (in-core-cool
       ((procedure
         ((c1 "CAUTION: If RWST level decreases to less than 40%,
the ECCS should be aligned for cold leg recirculation. Use
Guidelines, TRANSFER TO COLD LEG RECIRCULATION")
          (c2 "CAUTION: RHR pumps should not pump water greater
than 200 F without CCW to the RHR system")
          (1 "1. Verify ECCS Valve Alignment"
           ("1.1 STATUS - PROPER EMERGENCY ALIGNMENT" (confirm)
"Manually align valves as necessary" (confirm)))
          (2 "2. Verify ECCS Flow In All Trains:"
           ("2.1 CCP injection flow indicators - CHECK FOR FLOW
SI pump flow indicators - CHECK FOR FLOW
RHR pump flow indicators - CHECK FOR FLOW" (confirm) "Start pumps and
align valves as necessary.
Try to establish charging via PD charging pump from the RWST or VCT:
1) Reset SI, Containment Isolation Phase A and B.
2) Realign charging system and start PD charging pump." (confirm)))
          (3 "3. Check RCP Support Conditions"
           ("3.1 Status - AVAILABLE" (confirm) "Follow plant
specific guidelines" (confirm)))
          (4 "4. Check Accumulator Isolation Valve Status:"
           ("4.1 Power to isolation valves - AVAILABLE" (confirm)
"Restor power to isolation valve(s)" (confirm))
            ("4.2 Isolation valves - OPEN" (confirm) "Open isolation
valve(s) unless closed
```

```
after accumulator discharge" (confirm)))
            (5 "5. Check Core Exit TCs:"
             ("5.1 Core exit TCs - LESS THAN 1200 F" (confirm) (6))
             ((return)))
            (6 "6. Check Containment Hydrogen Concentration:"
             ("6.1 Obtain a hydrogen concentration measurement from:
1) Containment hydrogen monitoring system
-or-
2) The PASS system" (confirm))
              ("6.2 Hydrogen concentration - LESS THAN 6.0% IN DRY
AIR" (confirm)
               "Consult TSC staff for additional recovery actions"
(7))
              ("6.3 Hydrogen concentration - LESS THAN 0.5 IN DRAY
AIR" (confimr)
               "Place containment hydrogen recombiner inservice"
(confirm)))
            (c3 "CAUTION:  Alternate water source for AFW
pumps will be necessary if CST level decreases to less than 10%")
            (c4 "CAUTION:  A faulted or ruptured SG should
not be used in subsequent steps unless no intact SG is available")
            (7 "Check Intact SG Levels:"
             ("7.1 Narrow range level - GREATER THAN 10% (25% FOR
ADVERSE CONTAINMENT" (confirm)
              "Increase total AFW flow to restore narrow range level
greater than 10% (25% for adverse containment).  Confirm total AFW
flow greater than 470 gpm" (confirm) (n2))
             ("7.2 Control AFW flow to maintain narrow range level
between 10% (25% for adverse containment) and 50%"
              (confirm)))
            (8 "Check RCS Vent Paths:"
             ("8.1 Power to PRZR PORV block valves - AVAILABLE"
(confirm) "Restor power to block valves" (confirm))
              ("8.2 PRZR PORVs - CLOSED" (confirm) "Manually close
PRZR PORVs. If any PORV cannot be closed, close its block valves"
(confirm))
              ("8.3 Block valves - AT LEAST ONE OPEN" (confirm)
               "Open block valve unless it was closed to isolate an
open PRZR PORV" (confirm))
              ("8.4 Reactor vessel head vents - CLOSED" (confirm)
"Closed reactor vessel head vent(s)" (confirm))
              ("8.5 PRZR vents - CLOSED" (confirm) "Close PRZR
vent(s)" (confirm)))
            (n1 "NOTE: Partial uncovering of SG tubes is acceptable
in the following steps")
            (9 "9. Depressurize All Intact SGs to 170 PSIG"
             ("9.1 Dump steam to condenser at maximum rate" (confirm)
               "Dump steam at maximum rate using intact SGs
atmospheric" (confirm))
             ("9.2 Check SG pressures - LESS THAN 170 PSIG" (confirm)
"SG pressure stable or increasing" (confirm) (7))
             ("9.3 Check RCS hot leg temperatures - AT LEAST TWO LESS
THAN 400 F" (confirm)
              (rule (temperature hot.leg1 decreasing 1)) (7))
             ("9.4 Stop SG depressurization" (confirm)))
```

```
            (10 "10. Check If Accumulators Should Be Isolated:"
              ("10.1 At least two RCS hot leg temperatures - LESS
THAN 400 F" (confirm) (16))
              ("10.2 Close all accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (11 "11. Stop All RCPs"
              ((confirm)))
            (12 "12. Depressurize All Intact SGs To Atmospheric
Pressure:"
              ("12.1 Dump steam to condenser at maximum rate"
(confirm) "Dump steam at maximum rate using SG atmospherics"
              (confirm)))
            (13 "13. Verify ECCS Flow:"
             ("13.1 CCP injection flow indicators - CHECK FOR FLOW
SI pump flow indicators - CHECK FOR FLOW
RHR pump flow indicators - CHECK FOR FLOW" (confirm) "Core exit TCs
greater than 1200 F" (confirm) (12)))
            (14 "14. Check Core Cooling:"
              ("14.1 Core exit TCs - LESS THAN 1200 F" (confirm)
(16))
              ("14.2 At least two RCS hot leg temperatures - LESS
THAN 350 F" (confirm) (12)))
            (15 "LOSS OF REACTOR OR SECONDARY COOLANT"
              ((loca 12)))
             (n2 "NOTE: Normal support conditions are desired but not
required for starting the RCPs")
            (16 "16. Check Core Exit TCs"
              ("16.1 Status - LESS THAN 1200 F" (confirm) "Start RCPs
as necessary until core exit TCs less than 1200 F"
               (confirm) "Core exit TCs greater than 1200 F and all
available RCPs running" (confirm-)
                "Open all PRZR PORVs and block valves" (confirm)
"Still bad" (confirm)
                "Open reactor vessel head vents and PRZR vents to
containment" (confirm)))
             (17 "17. Try to Locally Depressurize All Intact SGs To
Atmospheric Pressure:"
              ("17.1 Use SG atmosphere" (confirm)))
            (18 "18. Check If Accumulators Should Be Isolated:"
              ("18.1 RHR flow indicators - AT LEAST INTERMITTENT
FLOW" (confirm) (16))
               ("18.2 Close all accumulator isolation valves"
(confirm) "Vent any unisolated accumulator" (confirm)))
            (19 "19. Check If RCPs Should Be Stopped:"
              ("19.1 At least two RCS hot leg temperatures - LESS
THAN 350 F" (confirm) (20))
               ("19.2 Stop all RCPs" (confirm)))
            (20 "20. Verify ECCS Flow"
              ("20.1 CCP injection flow indicators - CHECK FOR FLOW
SI pump flow indicators - CHECK FOR FLOW
RHR pump flow indicators - CHECK FOR FLOW" (confirm) "1) Continue
efforts to establish ECCS flow
2) Try to establish charging" (16)))
            (21 "21. Check Core Cooling:"
```

```
                ("21.1 At least two RCS hot leg temperatures - LESS
THAN 350 F" (confirm)))
              (22 "22. LOSS OF REACTOR OR SECONDARY COOLANT"
                ((loca 12)))))))
        (atws
          ((procedure
             ((n1 "NOTE: Step 1-3 should be performed immediately
If reactor trip occurs, immediately go to E-0, REACTOR TRIP OR SAFETY
INJECTION, STEP 2")
              (1 "1. Perform Following Actions From Control Room:"
               ("1.1 Try to trip reactor manually" (confirm) "Try to
manually insert control rods" (confirm))
               ("1.2 Try to trip turbine manually" (confirm) "Try to
runback turbine" (confirm)))
              (2 "2. Check AFW Pumps Running"
               ("2.1 Motor driven pump breaker indicator lights - LIT"
(confirm) "Manually start pumps" (confirm))
               ("2.2 Turbine driven pump steam supply valves - OPEN"
(confirm) "Manually open valves"))
              (3 "3. Check AFW Valve Alignment:"
               ("3.1 AFW valves - PROPER EMERGENCY ALIGNMENT" (confirm)
"Manually open or close valves as appropiate"
                (confirm)))
              (4 "4. Check If the Following Trips Have Occured:"
               ("4.1 Reactor Trip" (confirm) "Try to trip locally"
(confirm))
               ("4.2 Turbine Trip" (confirm) "Try to trip locally"
(confirm)))
              (5 "5. Verify AFW Flow:"
               ("5.1 AFW flow indicators - CHECK FOR FLOW" (confirm)
"Perform actions 2 and 3 locally" (confirm)))
              (c1 "CAUTION: Charging pump miniflow valves must remain
open when RCS pressure is greater than pump shutoff head")
              (6 "6. Initiate Rapid Boration Of RCS To Obtain Adequate
Shutdown Margin:"
               ("6.1 Start charging pumps" (confirm))
               ("6.2 Align boration flow path" (confirm))
               ("6.3 Check RCS Pressure - LESS THAN ??? PSIG" (confirm)
                "Open PRZR PORVs, as necessary, until RCS pressure drop
to ??? PSIG" (confirm)))
              (7 "7. Verify Containment Ventilation Isolation"
               ((confirm) "Manually isolate containment ventilation"
(confirm)))
              (8 "8. Maintain Adequate Shutdown Marin"
               ((confirm)))
              (9 "9. REACTOR TRIP OR SAFETY INJECTION"
               ((reactor.trip 2)))))))
        (losp
          ((procedure
             ((n1 "NOTE: Steps 1 and 2 are to be performed
immediately")
              (1 "1. Verify Reactor Trip:"
               ("1.1 All rod bottom lights - LIT
All rod position indicators - ZERO
Neutron flux - DECREASING" (confirm-) (2))
```

```
            ("Manually trip reactor" (confirm) (atws 1))
            ("If no instrumentation available, dispatch personnel to
restore power to vital instrument busses" (confirm)))
          (2 "2. Verify Turbine Trip:"
           ("2.1 All turbine stop valves - CLOSED" (confirm)
"Manually trip turbine" (confirm)))
          (c1 "CAUTION: When power is restored to one ac emergency
bus immediately go to step 19 and evaluate plant recovery options.
If an SI signal exists or if SI is actuated during this guideline, it
should be reset.")
          (3 "3. Try to Restore Power to Any AC Emergency Bus:"
           ("3.1 Load ac emergency bus on diesel")
           ("3.1.1 Start diesel" (confirm) "Follow plant specific
guidelines for emergency start of diesel")
           ("3.1.2 Verify automatic loading on diesel" (confirm)
"Manually load diesel" (confirm) "Trip diesel" (confirm))
           ("3.2 AC bus loaded" (confirm) "Load bus on any other
power supply" (confirm))
           ("3.3 AC emergency power restored" (confirm-) (19))
           ("3.3.1 Dispatch personnel to locally restore ac power"
(confirm))
           ("3.3.2 Place following safeguards component switches in
PULL-TO-LOCK position:
a) Charging/SI pumps
b) High head SI pumps
c) Low-head SI pumps
d) Containment spray pumps
e) CCW pumps
f) Motor-driven AFW pumps
g) Containment fan coolers" (confirm)))
          (c2 "CAUTION: Essential service water pump will
automatically load on energized ac emergency bus to provide diesel
generator cooling")
          (4 "4. Verify RCS Isolation:"
           ("4.1 Pressurizer PORVs - CLOSED" (confirm) "RCS
pressure greater than 2335" (confirm)
            "Manually close PORVs" (confirm))
           ("4.2 Letdown isolation valves - CLOSED" (confirm)
"Manually close valves" (confirm))
           ("4.3 Excess letdown isolation valves - CLOSED"
(confirm) "Manually close valves" (confirm)))
          (5 "5. Place Following Valve Switches In CLOSED
Position and Dispatch Personnel to Locally Close Valves To Isolate
RPC Seals:"
           ("5.1 Close RCP seal return outside containment
isolation valve" (confirm))
           ("5.2 Close RCP seal injection outside containment
isolation valves" (confirm))
           ("5.3 Close RCP thermal barrier CCW return outside
containment isolation valve" (confirm)))
          (6 "6. Verify AFW Flow:"
           ("6.1 AFW flow indicators - CHECK FOR FLOW" (confirm)
            "1) Verify turbine-driven AFW pump steam supply valves
open
```

```
2) Verify proper emergency alignment of AFW valves" (confirm)
"Manually open valves" (confirm)))
            (7 "7. Check CST To Hotwell Isolation:"
              ("7.1 Condenser hotwell isolation valves - CLOSED"
(confirm) "Manually close valves" (confirm)))
            (8 "8. Check Steam Generator Isolation"
              ("8.1 Main steamline isolation valves - CLOSED"
(confirm) "Manually close valves" (confirm))
              ("8.2 Main steamline isolation bypass valves - CLOSED"
(confirm) "Manually close valves" (confirm))
              ("8.3 Blowdown isolation valves - CLOSED" (confirm)
"Manually close valves" (confirm)))
            (9 "9. Check For Secondary Integrity:"
              ("9.1 All steam generator pressures - APPROXIMATELY
EQUAL" (confirm)
                "1) Isolate AFW flow of faulted steam genertor
2) Isolate faulted steam generator steam supply to turbine-driven AFW
pump" (confirm)))
            (10 "10. Check For Primary To Secondary Integrity:"
              ("10.1 Condenser air ejector radiation - NORMAL"
(confirm)
                "If HIGH, continue with this guideline while trying to
identify and isolate faulted steam generator" (confirm))
              ("10.2 Steam generator blowdown radiation - NORMAL"
(confirm)
                "If HIGH, continue with this guideline while trying to
identify and isolate faulted steam generator" (confirm)))
            (c3 "CAUTION: A faulted steam generator
that is isolated should remain isolated throughout further recovery
actions.")
            (11 "11. Check Non-Faulted Steam Generator Levels:"
              ("11.1 Narrow range level - GREATER THAN ???" (confirm)
                "Maintain full AFW flow until narrow range level is
greater than ???" (confirm))
              ("11.2 Manually control AFW flow to maintain narrow
range level at ???" (confirm)
                "If narrow range level in one steam generator
continues to increase with AFW flow stopped,
1) Isolate AFW flow to faulted steam generator
2) Isolate faulted steam generator steam supply to turbine-driven AFW
pump" (confirm)))
            (12 "12. Check CST Level:"
              ("12.1 CST level - GREATER THAN ??? %" (confirm)
"Switch to alternate AFW water supply" (confirm)))
            (13 "13. Check DC Bus Loads:"
              ("13.1 Shed all large non-vital dc loads as soon as
practical" (confirm))
              ("13.2 Dispatch personnel to monitor DC power supply"
(confirm)))
              (c4 "CAUTION: In order to prevent injection of
accumulator nitrogen into the RCS and to prevent the reactor core
returning to criticality due to moderator temperature effects, DO NOT
reduce RCS pressure below ??? psig OR core exit TC temperature below
?? F")
```

```
          (n2 "NOTE: 1) The non-faulted steam generators should be
depressurized as quickly as possible BUT in a controlled manner so
that RCS pressure and temperature limits are not violated.  2)
Continue with step 15 of this guideline on after depressurization of
non-faulted steam generators in step 14 has been started")
          (14 "14. Depressurize Non-Faulted Steam Generators to
Minimize RCS Inventory Loss:"
          ("14.1 Manually open and throttle steam generator PORVs
to reduce RCS pressure to ??? psig" (confirm)
          "Locally open and throttle PORVs" (confirm))
          ("14.2 Manually control AFW flow to maintain steam
generator narrow range level at ??? %" (confirm)
          "If less than ??? %,
1) Maintain full AFW flow until narrow range level is greater than
???%
2) Maintain steam generator levels above top of U-tubes, then stop
steam generator depressurization" (confirm))
          ("14.3 Manually throttle steam generator PORVs to
maintain RCS pressure at ??? psig" (confirm)
          "Locally throttle PORVs" (confirm)))
          (15 "15. Check RCS Conditions During Secondary
Depressurization:"
          ("15.1 RCS Pressure - GREATER THAN ?? PSIG" (confirm)
          "Stop steam generator depressurization AND restore RCS
pressure to ?? psig" (confirm))
          ("15.2 Core exit TC - GREATER THAN ?? F" (confirm)
          "Stop steam generator depressurization AND restore RCS
pressure to ?? psig" (confirm)))
          (16 "16. Verify and Reset SI Signal:"
          ("16.1 Verify SI signal actuated" (confirm) "Manually
initiate SI" (confirm))
          ("16.2 Reset SI signal" (confirm)))
          (17 "17. Verify Containment Isolation Phase A:"
          ("17.1 Isolation phase A valves - CLOSED" (confirm)
"Manually close valves" (confirm)))
          (18 "18. Check If Recovery Can Be Initiated:"
          ("18.1 One AC emergency bus - POWER RESTORED" (confirm)
          "Continue to control RCS conditions and monitor plant
status
1) Check status of auxiliary boration systems
2) Check status of spent fuel cooling
3) Control RCS pressure and temperature" (14)))
          (19 "19. Maintain Stable RCS Pressure And Temperature
While Evaluating Recovery Options:")
          (20 "20. Select Recovery Option:"
          ("20.1 Check RCS subcooling - GREATER THAN ??? F"
(confirm) "LOSP with SI REQUIRED")
          ("20.2 Check pressurizer level - GREATER THAN 10%"
(confirm) "LOSP with SI REQUIRED")
          ("20.3 Verify that SI components have not automatically
actuated upon ac power restoration" (confirm)
          "LOSP with SI REQUIRED")
          ("20.4 LOSP WITHOUT SI REQUIRED"))))))
    (lohs
      ((procedure
```

```
            ((c1 "CAUTION: DO NOT proceed if there is uncontrolled
depressurization of all steam generators and total feed flow is less
than 470 gpm 2) If wide range level in any 3 SGs is less than 25% or
PRZR pressure is greater than or equal to 2335 psig due to loss of
secondary heat sink, RCPs should be tripped and Steps 5-11 should be
immediately initiated for bleed and feed 3) Feed flow should bot be
re-established to any faulted SG if a non-faulted SG is available")
            (1 "1. Check If Secondary Heat Sink Is Required"
              ("1.1 RCS pressure - GREATER THAN NON-FAULTED SG
PRESSURE" (confirm) "LOCA" (loca 1))
              ("1.2 RCS temperature - GREATER THAN 350 F" (confirm)
               "Try to put RHR System in service while continuing with
guide." (confirm)))
            (2 "2. Try to establish AFW Flow To at Least One SGs:"
              ("2.1 Check control room indications for AFW failure:
1) CST level
2) AFW pump power supply
3) AFW valve alignment" (confirm))
              ("2.2 Try to restore AFW flow:
1) Restore level in CST and start AFW pumps
2) Align SSW to suction of AFW pumps and start FW pumps
3) Restore power to MDAFW pumps and start MDAFW pumps
4) Start TDAFW pump
5) Align AFW valves as necessary" (confirm))
              ("2.3 Check total flow to SGs - GREATER THAN 470 GPM"
(confirm)
               "Dispatch operator to locally restore AFW flow" (3))
               ("2.4 Return to procedure and step in effect"
(confirm)))
            (3 "3. Stop All RCPs"
             ((confirm)))
            (4 "4. Check CCP Status"
             ("4.1 Status - AT LEAST ONE AVAILABLE" (confirm) (10)))
            (c2 "CAUTION: If offsite power is lost after
SI reset, manual action may be required to restart safeguards
equipment")
            (n1 "NOTE: When establishing Main Feedwater to SGs, at
least two SGs should be used")))))


))

COMMENT:  MITIGATE acts a user interface with the guideline knowledge
base

(defun mitigate (rslt)
  (prog (rslt1 mp line ans)
      (send *standard-output* :expose)
      (send *standard-output* :select)
      (setq rslt1 rslt)
      (setq last nil)
      (setq line 1)
      (terpri)
     label
      (if (equal rslt1 nil) (return))
```

```
      (setq mp (car (kbcall 4 (append '(the procedure of) (car rslt1)
'(is ?a)) 1)))
     label1
      (prog (current step step1)
            (setq step (cdr (get-line line mp)))
            (princ (car step))
            (terpri)
            (setq step (cdr step))
            (go label3)
        label2
            (setq step (next-line line mp))
            (if (equal step nil) (return))
            (setq line (car step))
            (setq step (cdr step))
            (princ (car step))
            (terpri)
            (setq step (cdr step))
        label3
             (setq current (append (car rslt1) (list (car (next-line
line mp)))))
            (setq step1 (car step))
            (if (equal step1 nil) (go label2))
            (setq ans (eval-mit-rou current step1))
            (cond ((neq ans nil)
                  (cond ((eql (length ans) 2) (setq rslt1 (list (list
(first ans)))) (setq line (second ans)) (go label)))
                    (setq line (first ans))
                    (go label1)))
            (setq step (cdr step))
            (go label3))))

COMMENT:  EVAL-MIT-ROU evaluates rules in mitigation kb

(defun eval-mit-rou (current step)
  (prog (step1 ans)
     label
      (setq step1 (car step))
      (cond ((equal step1 nil) (return)))
      (cond ((stringp step1) (princ step1) (terpri) (setq step (cdr
step)) (go label)))
      (cond ((equal step1 '(return))
             (return last)))
      (cond ((equal (first step1) 'rule)
             (setq ans (eval-rule (second step1)))
             (cond ((equal ans 1) (return)))
             (cond ((equal nk nil) (princ "Not Confirmed") (terpri)
                  (setq step (cdr step)) (go label)))
             (cond ((equal nk 'yes) (setq step1 '(confirm))))))
      (cond ((equal step1 '(confirm))
             (setq ans (y-n-other-p "Is this True ?"))
             (if (not (or (eql ans 't) (eql ans 'nil))) (return ans))
             (cond (ans (terpri) (return)) ((terpri)))
             (setq step (cdr step))
             (go label)))
      (cond ((equal step1 '(confirm-))
```

```
            (setq ans (y-n-other-p "Is this True ?"))
            (if (not (or (eql ans 't) (eql ans 'nil))) (return ans))
            (cond ((not ans) (terpri) (return)) ((terpri)))
            (setq step (cdr step))
            (go label)))
      (if (= (length step1) 2) (setq last current))
      (return step1)))

(defun y-n-other-p (&optional str)
  (prog (ans)
     label
      (setq ans (prompt-and-read :character str))
      (if (eql ans #\y) (return 't))
      (if (eql ans #\n) (return 'nil))
      (cond ((eql ans #\()
             (setq ans (b-make-list (string-trim "()" (prompt-and-
read :string "Special >("))))
            (return ans)))
      (go label)))

(defun b-make-list (str)
  (prog (list i j sstr istr bstr)
      (setq bstr " ")
      (setq list nil)
      (setq i 0)
      (setq j 1)
      (setq sstr (substring str i j))
     label
      (cond ((= j (length str))
             (cond ((not (equal (substring str (- j 1) j) " "))
                    (setq list (append list (list (string-symbol
sstr))))))
             (return list)))
      (setq j (+ j 1))
      (setq bstr istr)
      (setq istr (substring str (- j 1) j))
      (cond ((equal istr " ")
             (setq i j)
             (cond ((not (equal bstr " "))
                    (setq list (append list (list (string-symbol
sstr))))))
             (go label)))
      (setq sstr (substring str i j))
      (go label)))

(defun get-line (line list1)
  (prog (list2)
     label
      (setq list2 (car list1))
      (cond ((equal (first list2) line)
            (return list2)))
      (setq list1 (cdr list1))
      (go label)))

;;;; next-line retreives the line immediately after 'line
```

```
(defun next-line (line list1)
  (prog (list2)
     label
      (setq list2 (car list1))
      (cond ((equal (first list2) line)
             (setq list2 (cadr list1))
             (return list2)))
      (setq list1 (cdr list1))
      (go label)))

COMMENT:  Below defines windows and commands used for presenting
Mitigation procedures

(DW:DEFINE-PROGRAM-FRAMEWORK Mitigation
  :x 730
  :y 43
  :width 390
  :height 300
  :SELECT-KEY
  #\o
  :COMMAND-DEFINER
  T
  :COMMAND-TABLE
   (:INHERIT-FROM '("colon full command" "standard arguments" "input
editor compatibility")
    :KBD-ACCELERATOR-P 'nil)
  :STATE-VARIABLES
  ((task-list nil))
  :PANES
  ((TITLE-1 :TITLE :HEIGHT-IN-LINES 1 :REDISPLAY-AFTER-COMMANDS NIL)
   (COMMAND-MENU-1 :COMMAND-MENU :MENU-LEVEL :TOP-LEVEL)
    (PANE-1  :DISPLAY  :INCREMENTAL-REDISPLAY T  :REDISPLAY-FUNCTION
'nil)
   (INTERACTOR-1 :INTERACTOR :HEIGHT-IN-LINES 4))
  :CONFIGURATIONS
  '((DW::MAIN (:LAYOUT (DW::MAIN :COLUMN TITLE-1 COMMAND-MENU-1 PANE-
1 INTERACTOR-1))
     (:SIZES
      (DW::MAIN (TITLE-1 1 :LINES)
       (COMMAND-MENU-1 :ASK-WINDOW SELF :SIZE-FOR-PANE COMMAND-MENU-
1) (INTERACTOR-1 4 :LINES)
       :THEN (PANE-1 :EVEN)))))
  :top-level
  (dw:default-command-top-level :dispatch-mode :form-preferred))



(define-mitigation-command (com-sgtr :menu-accelerator "SGTR")
                           ()
  (mitigate '((sgtr))))

(define-mitigation-command (com-sgtr-contingencies :menu-accelerator
"SGTR Contingencies")
```

```
                    ()
  (mitigate '((sgtr-c))))

(define-mitigation-command (com-si-termination :menu-accelerator "SI
Termination")
                    ()
  (mitigate '((si-t-sgtr))))

(define-mitigation-command (com-loca :menu-accelerator "LOCA")
                    ()
  (mitigate '((loca))))

(define-mitigation-command  (com-reactor-trip  :menu-accelerator
"Reactor Trip")
                    ()
  (mitigate '((reactor.trip))))

(define-mitigation-command  (com-inadequate-core-cooling  :menu-
accelerator "Inadequate Core Cooling")
                    ()
  (mitigate '((in-core-cool))))

(define-mitigation-command (com-atws :menu-accelerator "ATWS")
                    ()
  (mitigate '((atws))))

(define-mitigation-command (com-losp :menu-accelerator "LOSP")
                    ()
  (mitigate '((losp))))

(define-mitigation-command (com-lohs :menu-accelerator "LOHS")
                    ()
  (mitigate '((lohs))))
```

VITA

Robert P. Martin, son of William Paul and Sally Martin, was born September 15, 1965, in Long Beach, California. He graduated in the top 1% from Marshall High School in Marshall, Texas. He received his Bachelor of Science Degree in Nuclear Engineering from Texas A & M University in May 1987. Mr. Martin can be reached at 405 Denise Drive, Marshall, Texas, 75670.