SRC-TR-87-59

CHEMICAL PROCESS SYSTEMS LABORATORY

A Method of Fault Diagnosis: Presentation of a Deep Knowledge System

by

D.T. Chung
M. Modarres

RESEARCH REPORT

CHEMICAL PROCESS SYSTEMS ENGINEERING LABORATORY

A METHOD OF FAULT DIAGNOSIS: PRESENTATION OF A DEEP KNOWLEDGE SYSTEM

D. T. Chung
M. Modarres

A CONSTITUENT LABORATORY OF
THE SYSTEMS RESEARCH CENTER

THE UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND 20742

A METHOD OF FAULT DIAGNOSIS: PRESENTATION OF A DEEP KNOWLEDGE SYSTEM

D. T. Chung

M. Modarres *

Department of Chemical and Nuclear Engineering

The University of Maryland

College Park, Maryland 20742

Presented for Publication to
Computers & Chemical Engineering Journal
(November, 1986)

* To whom correspondence should be addressed.

A METHOD OF FAULT DIAGNOSIS: PRESENTATION OF A DEEP KNOWLEDGE SYSTEM

D. T. Chung

M. Modarres

Department of Chemical and Nuclear Engineering

The University of Maryland

College Park, Maryland 20742

ABSTRACT

In fault diagnostic expert systems, the knowledge can be either shallow (experience-based) or deep (function-based). This paper presents a deep-knowledge expert system for fault detection in process operations and control domain. The structure used for modelling deep-knowledge is called Goal-Tree-Success Tree. An expert system shell has been built for applying this deep-knowledge model utilizing the logic based language PROLOG. An example, applying this deep-knowledge model and the developed expert system shell, is also presented.

A METHOD OF FAULT DIAGNOSIS: PRESENTATION OF A DEEP KNOWLEDGE SYSTEM

D. T. Chung

M. Modarrés

Department of Chemical and Nuclear Engineering

The University of Maryland

College Park, Maryland 20742

Scope

The operational complexity of large processing plants requires plant operators to handle many difficult and knowledge intensive tasks. Plant operations and diagnosis of faults in process plants are amenable to applications of artificial intelligence, specifically expert systems. Expert system technology has evolved to the point that a variety of general purpose advisory and diagnostic systems are available to aid operation and control.

Currently there are two types of methods for modelling knowledge for expert system applications. These two types are shallow knowledge and deep-knowledge. For complex process plants shallow knowledge models are difficult to formulate and are often incomplete. In this paper, we will present an approach for building expert system for plant

diagnostic that uses a deep-knowledge model based on the plant operational goals. An expert system building shell and an example of its application is provided. This expert system is currently available for off-line applications. On-line real-time development of the system is underway.

A METHOD OF FAULT DIAGNOSIS: PRESENTATION OF A DEEP KNOWLEDGE SYSTEM

D. T. Chung

M. Modarrés

Department of Chemical and Nuclear Engineering

The University of Maryland

College Park, Maryland 20742

Conclusions and Significance

In this paper we have discussed the development of an expert system shell (called GOTRES) based on a deep knowledge model called Goal Tree-Success Tree (GTST) model. The GOTRES expert system demonstrated that deep knowledge for process control plants can be effectively represented by using the GTST model, and the model can be conveniently arranged into frame-based knowledge by using the language PROLOG. It is observed that GOTRES shell can quickly process the frame-based knowledge and assist the user to locate the site of fault(s) when a goal is lost. It is also concluded that by incorporating real-time interface into the GOTRES, this expert system will be capable of automatically acquiring plant data which will enhance the GOTRES capabilities.

1. INTRODUCTION

The purpose of a fault diagnostic expert system in a process control environment is to aid the operator in detecting and resolving process failures. Two general approaches have been applied to the development of expert systems; model-based (deep-knowledge) and model-free (shell knowledge).

A survey of shallow knowledge fault detection expert systems has been discussed by Pau [2], and Waterman [3]. Shallow knowledge expert systems use (if-then) type rules as the primary mean of knowledge representation. These rules are formulated based on a large collection of empirical observations. Typically, the development steps for a rule-based system can be generalized as: data abstraction, heuristic mapping onto a hierarchy of pre-enumerated solutions, and refinement within this hierarchy [3].

Performance of these rule-based diagnostic expert systems can become very effective, provided that all failure modes have been compiled, and the failures can be sufficiently characterized from measurements and/or observations. In cases where the failure modes are not well known, these systems are inadequate, and deep knowledge systems are more appropriate.

When confronted with an unfamiliar problem an expert can resort to "first principles". Through in-depth understanding of the problem, an expert can resolve problems that have not been well documented by prior observations [4]. For

example, in fault detection in a process control domain an expert can diagnose the failure through his fundamental understanding of the principles behind the operations of the plant. In electronics, an expert may isolate the fault by tracing through the functional structure of an electronic circuit [5,6]. The knowledge used by the expert in these situations are referred to as "deep knowledge".

There are numerous diagnostic expert systems using deep knowledge. A number of these systems are presented in the literature [6-12]. An example of a deep knowledge expert system is IBM's Diagnostic Assistance Reference Tool (DART) [11]. This system is designed to diagnose faults in computer hardware. The DART system contains no information about how computer hardware fail. Instead, has a structural description of the computer hardware. It works directly from information about intended structure (machine parts and their interconnections) and expected behavior (equations, rules, or procedures that relate inputs and outputs). This approach provides a broader knowledge base, as well as modularity for incorporating new knowledge. The information needed for DART's knowledge base can be accumulated as the device is designed. DART uses the structural information to deduce a fault.

In process control, there is a large spectrum of conditions that can lead to failure. Many of these failures are unanticipated and have very low probability of occurring. As a result they are not thoroughly characterized and cannot

be represented through rule-based shallow knowledge. However, these failure conditions can be diagnosed by using fundamental understanding of the principles behind the operation of the plant. Hence, deep knowledge methods can be more effective in process operation and diagnosis domain. While DART uses the structural functions as the deep knowledge; it can perform diagnosis when system is not in operation during which input-output checks can be performed. Deep knowledge method based on structural functions would not work as well in complex process control where a system cannot be shutdown while diagnostic tests are being performed on the system. An effective deep-knowledge method which requires no plant shutdown during the diagnose would have to be used in process plant. Since various instruments are used to monitor and determine whether process plant hardware are working, and since plant hardware support specific plant goals (or functions), a goal oriented method to represent the deep-knowledge would be more effective.

The purpose of this paper is to present an expert system that utilizes a deep-knowledge model that is based on specific plant goals. In this method the principles of process operations and control are organized in a tree format. This method of modelling knowledge is known as Goal Tree-Success Tree (GTST) [13,14].

In this paper we will discuss development of an expert system shell based on the GTST model. The intelligent language PROLOG [15-17] is used to program this shell. In its

present form, GTST modelled knowledge and instrument readings are required for determining the site of faults. These data are entered into the shell by the user. Currently efforts are underway to upgrade the system so that the instrument readings are obtained directly through a real-time data acquisition interface. In the rest of the paper the methodology along with an application of the approach is presented.

2. The GTST METHODOLOGY

The knowledge base used in this expert system is based on the GTST model. The GTST model is a hierarchical representation of plant goals and hardwares. This top-down hierarchical model represents a collection of goals and sub-goals that are required to achieve a pre-defined plant objective. There are specific rules for developing the GTST model so as to maintain hierarchy and completeness [13]. The first step in development of the tree involves definition of the top goal or objective. This top goal must be explicitly defined in terms which make it a single unambiguous statement. It is from this definition that the analyst will identify and relate all the different plant goals and sub-goals which must be achieved to attain the overall objective.

The goal tree is built vertically downward from the objective in levels, wherein the analyst subsequently decomposes each identified goal into a necessary and sufficient set of dependent sub-goals. As the vertical detailed development of the tree increases, it is necessary

that specific rules for the GTST model development be applied to ensure its accuracy and completeness; and that the proper hierarchy between goals and sub-goals be rigorously maintained. Important rules during the GTST development are:

- Upon looking downward from any goal towards the bottom of the tree, it is possible to define explicitly how the specific goal or sub-goal is satisfied.
- Upon looking upward from any sub-goal towards the objective or top-goal, it is possible to define explicitly why the specific goal or sub-goal must be satisfied.

When the goals are sufficiently broken down into lower level sub-goals there will be a level beyond which sub-goals can only be defined with reference to actual hardware. As soon as hardware is explicitly mentioned, the tree becomes one which describes "success paths". A success tree shows the combination of components that should work so that the hardware works. Figure 1 shows a typical GTST model.

3. EXPERT SYSTEM SHELL

The goal tree expert system (GOTRES) shell consists of a knowledge base and an inference engine. In the current configuration there is also an interactive front-end program for entering plant status through the instrument readings.

In the following, all elements of the GOTRES shell are described.

(1) Knowledge Base

The knowledge base of GOTRES contains the attributes associated with each goal and the structural information of the GTST. The structure of the GTST is entered into the computer using the frame-based representation method [1]. In this method, the goal tree is viewed as a semantic network. Where the goals are nodes in the semantic net; and they are connected using the relational connector "how". This network can be seen in Fig. 1.

Once the frame-based representation has been established, then PROLOG fact statements can be written to represent the semantic network and the attributes of the goals. PROLOG is particularly suitable to show semantic networks and attributes since it can easily show relations between items. For those readers unfamiliar with the relational aspect of PROLOG, this process involves defining PROLOG predicate, and arguments. For example, in order to enter the semantic network into the knowledge base one would treat the connector "how" as a predicate. The two goals that are being connected are the arguments of this predicate.

Suppose goal Y, according to the GTST model, is achieved by goal X, the PROLOG fact statement would represent this as:

((how (X) (Y))).

In this case X is a direct subgoal of Y. As will be seen in the later example, there is a "how" fact statement for every connection in the goal tree. Similarly, the connections between the hardwares and the lowest level goals are entered

into the knowledge base using this type of fact statement. If goal X is a lowest level goal and hardware Z is required for accomplishing X, the fact statement can be shown by:

((how (Z) (X))).

To differentiate between the goals and hardware, information defining their characteristics must be included as PROLOG fact statements. For example, the following fact statement shows that Z is a hardware.

((hardware (Z))).

For each of the goals in the GTST model, four fact statements are entered in the knowledge base. Three of these are for identifying the features of the goal, and one is for indicating the conditions for goal success or failure. The first of these statements is:

((goal-num n)).

This indicates that integer "n" is a goal number. By having goal numbers in the knowledge base the users will not have to type lengthy goal statements.

The second of these statements is:

((goal-statement (n) (X))).

The two arguments in this statement are the goal number and the statement of goal X. This fact statement declares that X is the goal statement of goal number "n". The parenthesis around X indicates that X can be a list of words.

The third of these statements is :

((goal-attribute (n) (c))).

The two arguments here are the goal number and detail statement describing the goal.

The fourth of these statements is:

((goal-instrument-support (n) (a b c))).

The arguments are, again the goal number, follow by a list of indicators that would have to be "proper" for the goal to be successful. Although the statement allows for a list, it can be a single indicator as well.

All of the four statements above contain the same integer goal number. This common denominator links the statements to the same goal, and similar set of PROLOG fact statements exist for each goal. Specific examples of these four statements will be later discussed in an example.

(2) Interactive Front-End

The interactive front-end program is designed to gather the instrument readings and plant conditions needed to evaluate goal and hardware success or failure. This program also acquires from the user whether each instrument is in the proper operating range, and stores the information in the knowledge base as PROLOG predicate statement. Using these predicate statements and the GTST information in the knowledge base, the inference engine part of the expert system is able to infer the possible causes of failure. A typical session of this program will also be seen later in an example. The front-end portion of this expert system shell is designed such that when real-time interaction is established, most of the data such as instrument reading can be directly obtained through data acquisition link. Some of the more essential PROLOG control rules of this program are explained in more detail in Appendix A.

(3) Inference Engine

The inference engine provides control strategy for processing the knowledge stored in the knowledge base, and infers possible causes of faults. The control strategy used by GOTRES in this process is backward chaining depth search. This search process identifies all branches of the GTST as successful or lost while checking the goals. Once a lost goal is determined, the search goes one level lower. Therefore, on the next lower level of the GTST only sub-goals of the higher level lost goal are checked. As a result, in this search process the inferece engine minimizes the number of goals needed to be checked before locating the lost goal.

To start a search, the user enters the command "exam" followed by an indication of the goal from which the search initiates. For example, if a user would want to start from the top goal. This can be done by entering the top goal statement. However, since integer goal numbers are installed in the knowledge base, this search can be initiate by just entering "exam 1". Here, "1" is the top goal number. Similarly, the user can initiate the search from any other goal in the GTST model.

Beginning with the top goal, the process checks the goal requirements for success, and compares them with the list of instrument indications entered through the front-end program. This checking process is performed by PROLOG control rules using the recursion process in the PROLOG. For the top goal in the GTST to be successful, all lower level

goals must be successful. If the top goal is lost, then at least one of the lower level goals must have been lost. The diagnostic process records the goals that are checked, and follows the failed goal to lower level goals. On each descending level, the goals are checked until a lost goal is identified. Once a lost goal is identified, the checking process continues depth-wise until reaching the lowest level lost goal in the GTST model. The actual hardware supporting a lowest level lost goal are then listed as possible sites of failures. The expert system then goes back to the initial lost goal and repeats the checking process by pursuing another failure path, if any, and present other possible failures. A graphical display of this search strategy is presented in Appendix B.

4. EXAMPLE

Nitric acid cooler is an example of a very simple process control system, see Fig.2. For this system, the objective is to cool nitric acid (HNO3) through a heat exchanger to within a designed temperature range. The control mechanism of this system is also modeled. For major control of coolant flow, a speed controller is used; while for fine control of the flow a temperature controller is used. In order to build a knowledge base for this system, one should first formulate a GTST model of the process. By applying the rules described earlier, a GTST model has been developed for this process as shown in Fig.3. This GTST model represents the basic principles for the operation and

control of the nitric acid cooler. Structure of the GTST, arranged in the form of frame-based representation, is placed into the knowledge base using PROLOG relational statements. These PROLOG statements use the predicate "how" as the structural connector. All of the PROLOG fact statements required to enter the GTST into the knowledge base are listed in Fig. 4.

Each goal in the GTST is represented by four PROLOG fact statements. For example, statements for the objective in the nitric acid cooler are as follows:

```
((goal-num 1)).
```

This (first) statement indicates that the goal number "1" is a valid goal number.

```
((goal-attribute (1)
    (This is the objective of the goal tree)).
```

This (second) statement provides a brief explanation of the top goal.

```
((goal-statement (1)
    (cool HNO3 to within designed temperature range))).
```

This (third) statement is the goal statement as it appears in the GTST model.

```
((goal-instrument-support (1) (T2) ))
```

This (fourth) statement indicates that for goal 1 to be successful temperature sensor 2 must be in the proper operating range.

There is a set of PROLOG fact statements indicating whether each sensor is in the proper range for the process

control operation. These fact statements are entered into the knowledge base using the interactive front-end program. For each sensor, the interactive front-end program inquires the user for the reading. A session with the interactive front-end program is shown in Fig 5. The interactive program is activated by the command "check hno3". User of the interactive program indicates whether the instrument is in the proper operating range by simply answering yes or no.

Evaluation of the facts in the knowledge base and the instrument values entered by the user is initiated by the command "exam 1". The diagnostic result is shown in Fig. 6. This inference engine also permits one to easily extract information from the knowledge base. For example, to receive the goal-attribute of a goal, the user needs to enter the predicate followed by the goal number. In this case, if the user enters,

"attribute 1"

the response would be:

For goal 1, the attribute is: (This is the top objective in the goal tree)

This feature of the GOTRES is further illustrated in Fig.7.

ACKNOWLEDGEMENT

The authors wish to acknowledge the support of the University of Maryland's System Research Center during the course of the work described in this paper.

REFERENCES

[1] P.H.Watson,´Artificial Intelligence´, Addison-Wesley,1983.

[2] L.F. Pau,´Survey of Expert System for Fault Detection, Test Generation and Maintenance´ Expert Systems,Vol.1, April 1986.pp.100-111.

[3] D.A.Waterman,´A Guide to Expert Systems´,Addison-Wesley Publishing Company,1986.

[4] J.S. Kowalik,´Knowledge Based Problem Solving´, Prentice-Hall,1986.

[5] B.Chandrasekaran, S.Mittal,´Deep Versus Compiled Knowledge Approaches to Diagnostic Problem-Solving´, Proc. AAAI-82, 1982,PP.349-354.

[6] R. Davis, H. Shrobe, W. Hamscher, K.Wieckert, M.Shirley, S.Polit,´Diagnosis Based On Description of Structure And Function´, Proc. AAAI-82, 1982, pp.137-142.

[7] W. Hamscher, ´Using Structural and Functional Information In Diagnostic Design´, Proc. AAAI-83, 1983, pp.152-156.

[8] R.Milne, ´Fault Diagnosis Through Responsibility´, Proc. 9th IJCAI,1985, pp.423-25.

[9] T.J. Laffey, W.A. Perkin, and T.A. Nguyen,´Reasoning About Fault Diagnosis with LES´,IEEE Expert,Vol.1, Spring 1986. pp.13-20.7

[10] R.Davis,´Reasoning From First Principles In Electronic Troubleshooting´,Int.J. Man-Machine Studies,1983, vol.19,pp.403-423.

[11] M.R. Genesereth, ´Diagnosis using hierarchical design methods´, Proc. AAAI, August 1982, pp.278-83.

[12] M.A.Kramer,B.L.Palowitch Jr.,´Expert System and Knowledge-based Approaches To Process Malfunction Diagnosis´AIChE National Meeting, Chicago, Nov. 1985.

[13]T.Cadman, M.Modarres,´A Method of Alarm System Analysis In Process Plants With The Aid of An Expert Computer System´,presented for publication to Computers & Chemical Engineeering Journal, Sept. 1985.

[14] R.N.Hunt, M.Modarres, M.L.Roush,´Application of Goal Trees to Evaluation of The Impact of Information Upon Plant Availability´, Proc. ANS/ENS topic meeting on probabilistic safety methods and applications, San Francisco, CA, Feb.1985.

[15] K.L.Clark, F.G.McCabe,´Micro-Prolog:Programming In Logic´,Prentice/Hall International, 1984.

[16] H.de Saram,´Programming In Micro-Prolog´,Ellis Horwood Limited,1985.

[17] D.Li,´A Prolog Database System´,Research Studies Press,1984.

APPENDIX A. PROLOG PROGRAMMING ASPECTS OF GOTRES

I.Fact Statements Used In the Knowledge Base

FACT	EXPLANATION
((how (X) (Y)))	/X is how Y is accomplished. This connects X to the higher goal Y.
((how (Z) (X)))	/Z is how Y is accomplished. This relates X to its sub-goal Z.
((hardware Z))	/Z is hardware item. This statement exists for each hardware.
((goal-num n))	/n is a goalnumber.
((goal-statement (n) (X)))	/n is the node number of goal X.
((goal-attribute (n) (c)))	/c is an explanation describing goal X.
((goal-instrument-support (n) (a b c)))	/a b c represent the instruments that can verify success of goal n.

II. Examples of Interactive Front-end Program Control Rules

RULE	EXPLANATION
((scan Z (X|Y)) ((PP Is X in the acceptable operating range?) (PP Please type y for yes or n for no.) (R Y1) (eval Y1 X) (scan Z Y))	/Recursion permits the inquiring of all instrument states. Response is read as Y1, and eval rule is called. (eval rule is explained below)
((eval Y X) (IF (EQ Y y) ((ADDCL((proper X)))) ((evaltwo Y X))))	/Response is carried as Y. If the response is yes, then the fact statement ((proper X)) is added to the knowledge base. If not evaltwo rule is called.
((evaltwo Y X) (IF (EQ Y n) ((ADDCL((fail X)))) ((ABORT))))	/If the response is no the fact statement ((fail X)) is added. Only y and n are accepted.

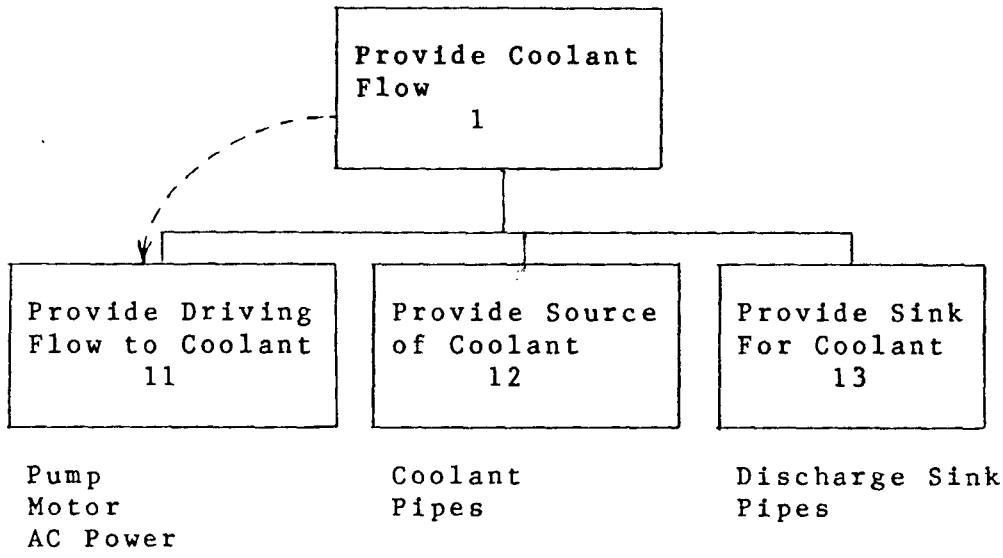
III. Examples of Essential Inference Engine Control Rules

RULE	EXPLANATION
((test x ()) (PP goal x is successful))	/if the list of instrument for x is empty, then goal x is successful.
((test x (Z|X)) (proper Z) (test x Y))	/Z is first in the instrument list. If it is proper then check the next item in the list.
((test x (Z|X)) (!trim x Z))	/instrument Z is not in the proper range. Goal x failed.

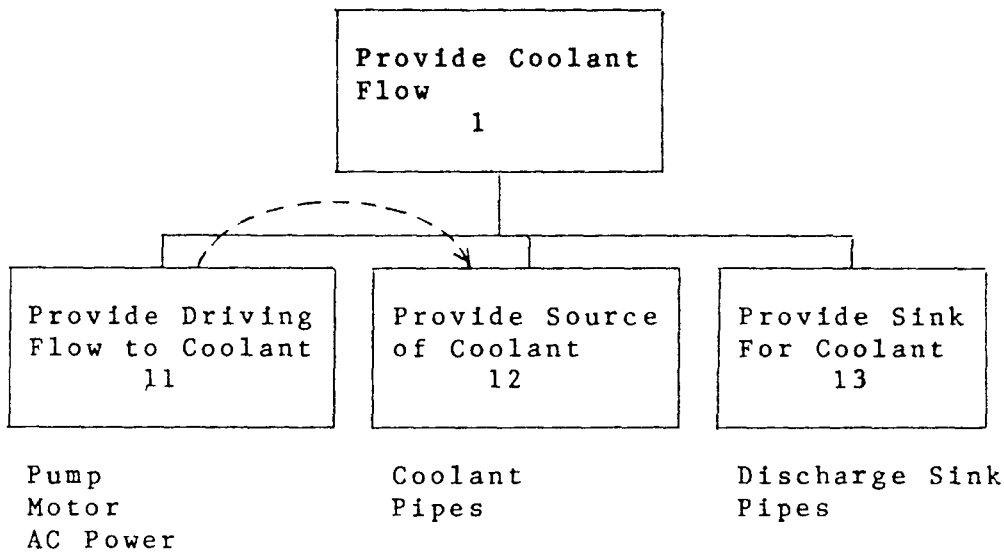
In this rule, the instrument readings associated with goal x is being tested. This collection of PROLOG statements perform recursion. In the recursion process, a list of items are tested one at a time.

RULE	EXPLANATION
((check x Y) (how Y1 X) (IF (hardware Y1) ((show Y)) ((checkon x Y))))	/checks if goal x is the lowest level goal. If x is connected by "how" predicate to hardwares, then hardware is shown. Otherwise, checking is continued.
((show X) (PP The following are possible cause of failure) (FORALL ((how X1 X)) ((PP X1))))	/prints all hardware that are connected to the sub-goal X.

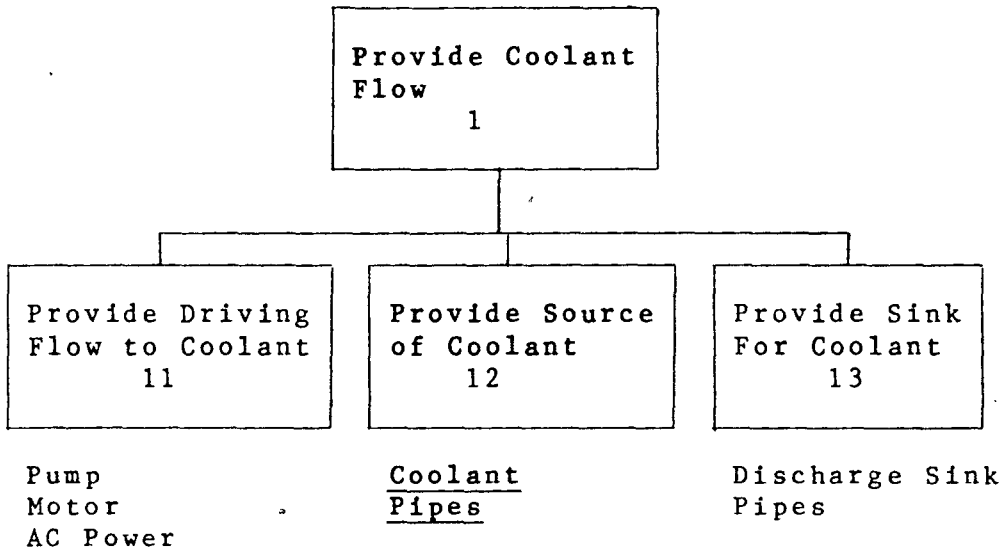
APPENDIX B. SEARCH STRATEGY USED BY GOTRES



(1) The objective of the GTST has failed, the search continues on the next lower level of the tree.



(2) Goal 11 is successful, and the search continues on the same level for a lost goal.



(3) The search identifies goal 12 as the lost goal; thus the hardware referenced by goal 12 will be listed as possible faults.

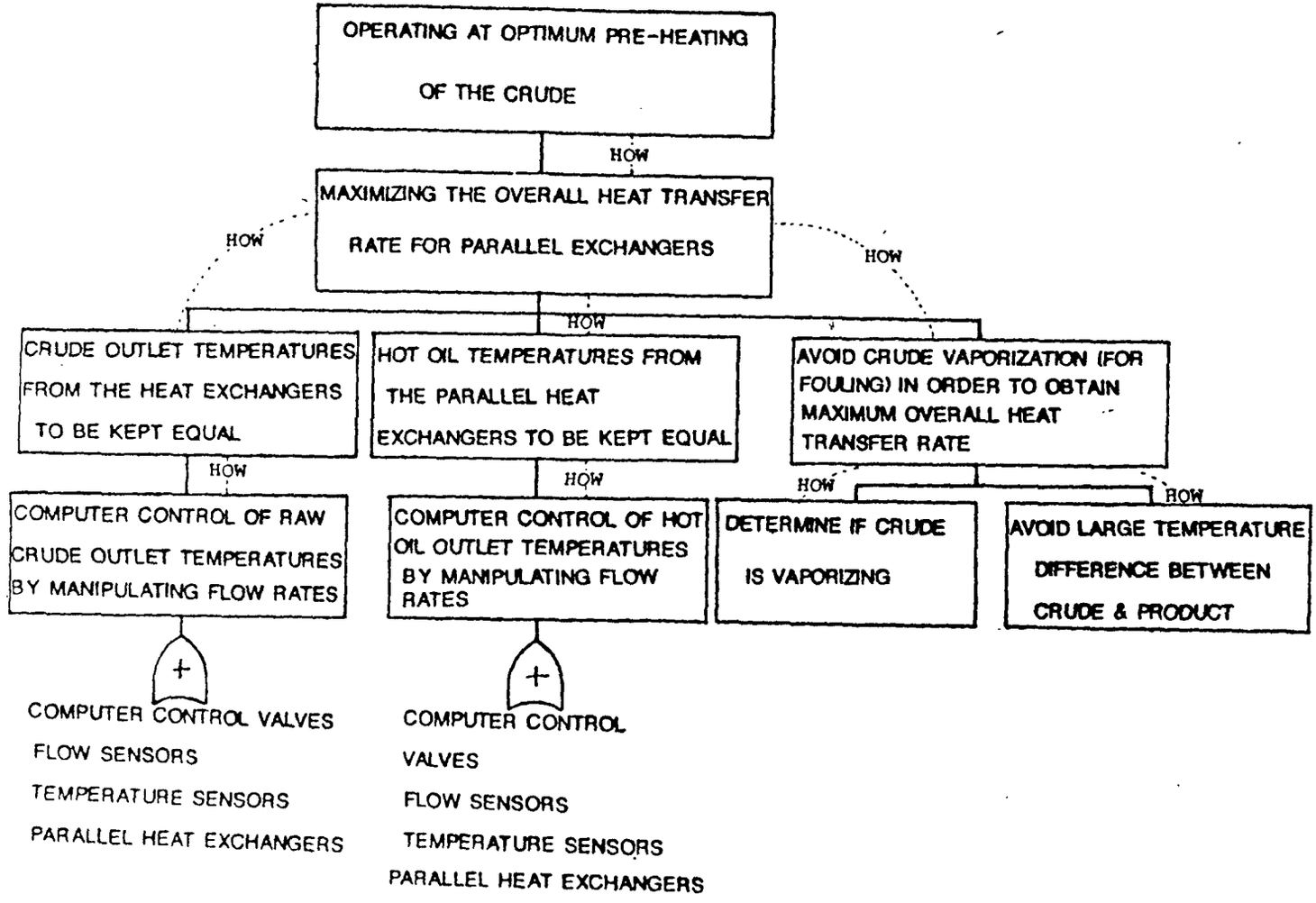


Figure 1. A typical Goal Tree-Success Tree.

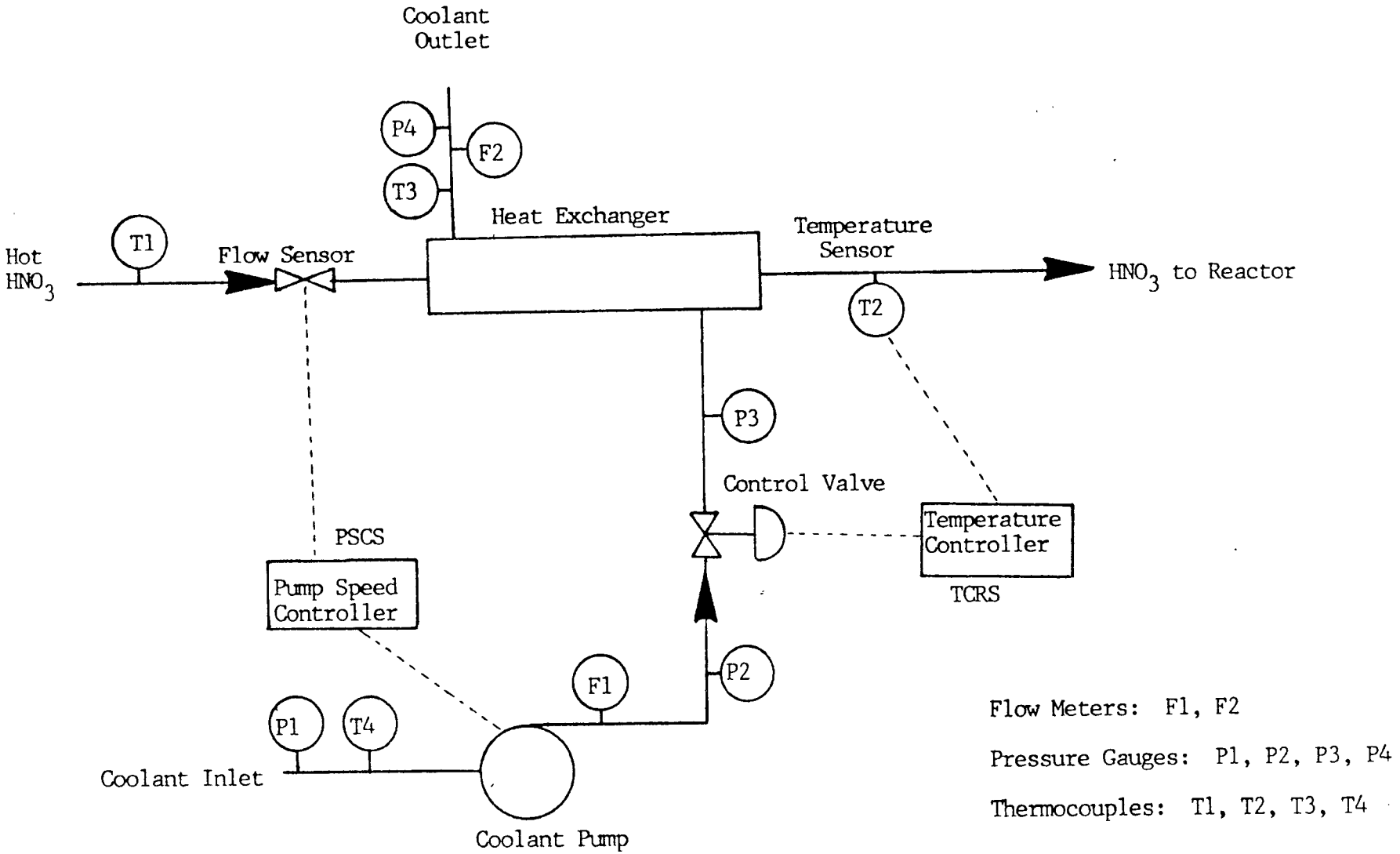


Figure 2. Nitric Acid Cooler System.

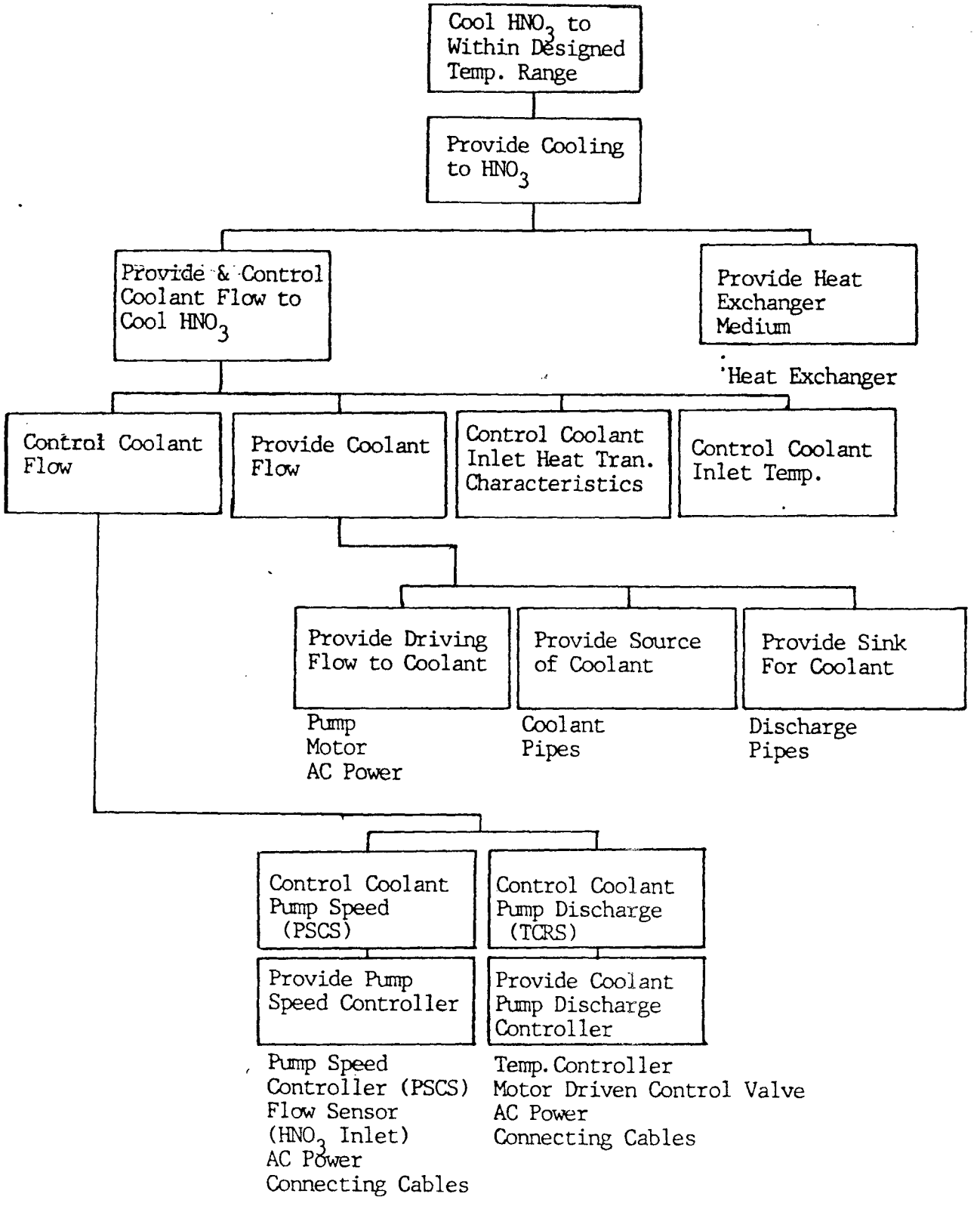


Figure 3. Goal Tree - Sucess Tree for the Nitric Acid Cooler System.

```
list how
((how (coolant temperature characteristics) (control coolant inlet heat transfer characteristics)))
((how (coolant temperature) (control coolant inlet temperature)))
((how (provide cooling to HNO3) (cool HNO3 to within designed temperature range)))
((how (provide and control coolant flow to cool HNO3) (provide cooling to HNO3)))
((how (provide heat exchanger medium) (provide cooling to HNO3)))
((how (control coolant flow) (provide and control coolant flow to cool HNO3)))
((how (provide coolant flow) (provide and control coolant flow to cool HNO3)))
((how (control coolant inlet temperature) (provide and control coolant flow to cool HNO3)))
((how (control coolant pump speed) (control coolant flow)))
((how (provide pump speed controller) (control coolant pump speed)))
((how (control coolant pump discharge) (provide coolant pump discharge controller)))
((how (pump speed controller) (provide pump speed controller)))
((how (flow sensor) (provide pump speed controller)))
((how (AC power) (provide pump speed controller)))
((how (connecting cables) (provide pump speed controller)))
((how (Temperature controller) (provide coolant pump discharge controller)))
((how (motor driven control valve) (provide coolant pump discharge controller)))
((how (AC power) (provide coolant pump discharge controller)))
((how (connecting cables) (provide coolant pump discharge controller)))
((how (provide driving flow to coolant) (provide coolant flow)))
((how (provide source of coolant) (provide coolant flow)))
((how (provide sink for coolant) (provide coolant flow)))
((how (pump) (provide driving flow to coolant)))
((how (motor) (provide driving flow to coolant)))
((how (AC power) (provide driving flow to coolant)))
((how (coolant) (provide source of coolant)))
((how (pipes) (provide source of coolant)))
((how (discharge sink) (provide sink for coolant)))
((how (pipes) (provide sink for coolant)))
((how (heat exchanger) (provide heat exchanger medium)))

((how (control coolant inlet heat transfer characteristics)
      (provide and control coolant flow to cool HNO3)))
&
```

Figure 4. "how" predicate statements used for representing the GTST.

```
 check hno3
Is T1 in the acceptable operating range ?
Please type y for yes , or n for no .
y
Is T2 in the acceptable operating range ?
Please type y for yes , or n for no .
n
Is T3 in the acceptable operating range ?
Please type y for yes , or n for no .
y
Is T4 in the acceptable operating range ?
Please type y for yes , or n for no .
y
Is P1 in the acceptable operating range ?
Please type y for yes , or n for no .
y
Is P2 in the acceptable operating range ?
Please type y for yes , or n for no .
n
Is P3 in the acceptable operating range ?
Please type y for yes , or n for no .
n
Is P4 in the acceptable operating range ?
Please type y for yes , or n for no .
n
Is F1 in the acceptable operating range ?
Please type y for yes , or n for no .
n
Is F2 in the acceptable operating range ?
Please type y for yes , or n for no .
n
The reading of indications are completed :
proper T1
fail T2
proper T3
proper T4
proper P1
fail P2
fail P3
fail P4
fail F1
fail F2
?
&
```

Figure 5. An interactive session with the front-end program.

```
 exam 1

goal number : 1

1 has failed because
T2 is a not in the proper operating range .

11 has failed because
T2 is a not in the proper operating range .

111 has failed because
T2 is a not in the proper operating range .

1111 has failed because
T2 is a not in the proper operating range .

11111 has failed because
T2 is a not in the proper operating range .

111111 has failed because
T2 is a not in the proper operating range .
the following are possible cause of failures
(pump speed controller)
(flow sensor)
(AC power)
(connecting cables)
```

Figure 6. The result of a diagnostic session.

```
&attribute 1
For goal 1, the attribute is:
(This is the top objective in the goal tree.)

&statement 1
1 (cool HNO3 to within designed temperature range.)

&attribute 1111
For goal 1111, the attribute is:
(The controlling of the amount of cooling is conducted by
adjusting the coolant flow rate.)

&statement 1111
1111 (control coolant flow)

&instrument 11
The instruments that are/is required to be in the correct
operating range for goal 11 to be consider a successful: (T2
T3 P1 P2 P3 F1 F2)

&instrument 111111
The instruments that are/is required to be in the correct
operating range for goal 111111 to be consider a successful:
(T2 T3)
```

Figure 7. Examples of information extraction from the knowledge base.