Second Annual Symposium, Mary Kay O'Connor Process Safety Center
"Beyond Regulatory Compliance: Making Safety Second Nature"
Reed Arena, Texas A&M University, College Station, Texas
October 30-31, 2001

Computer-aided Fault Tree Synthesis for Quantitative Risk Assessments

Yanjun Wang,
T.L. Teague, Harry West, and M. Sam Mannan
Mary Kay O'Connor Process Safety Center
Chemical Engineering Department Texas A&M University
College Station TX 778433122

Phone: 979/862-3985
Email: yjwang@tamu.edu

ABSTRACT

Fault tree analysis (FTA) has been used in the chemical process industry (CPI) for systematic safety and reliability analysis during the past decades. Once constructed, the fault tree can be of considerable value in determining the paths for propagation of basic events through the system to cause the top event. Algorithms exist that determine which basic events, or combinations of primal events (minimum cut set analysis), will cause the top event for a given fault tree. While much of the statistical and cut set analysis has been automated, construction of a fault tree is usually done by hand. Manual construction of the tree can be extremely time consuming and vulnerable to human errors. No entirely-satisfactory algorithm has been published for fault tree synthesis, especially when control loops are encountered.
Ideally the system failure models should be independent of the synthesis method, but in practice they are strongly interdependent. The object of this research is to develop an efficient and practical computer algorithm for the synthesis of fault trees. The fault tree is deduced based on a mini-fault tree model of the analyzed system. The correctness of this algorithm will be examined for control loops. Serial and parallel consistency also will be considered and discussed.

A New Algorithm for Computer-aided Fault Tree Synthesis

Yanjun Wang, Tom Teague, Harry West, and M. Sam Mannan
Mary Kay O'Connor Process Safety Center
Chemical Engineering Department
Texas A&M University
College Station, TX 77843-3122

Abstract

Fault tree analysis (FTA) has been used in the chemical process industry (CPI) for systematic safety and reliability analysis during the past decades. Because manual construction of fault trees can be extremely time consuming and vulnerable to human error, automation of fault tree synthesis is highly desirable. However, no entirely satisfactory algorithm has been published for fault tree synthesis, especially when control loops are encountered.

A new methodology to construct fault trees automatically is proposed in this paper. System block diagram and cause and effect unit models are employed to model chemical processes. An example is embedded in the description of the methodology for better understanding. Analysis shows that the fault tree generated here is equivalent to the published result.

This algorithm works directly form the system block diagram, thus avoids the tedious work of drawing digraphs, transition tables, *etc.* Control loops are considered and treated by special cause and effect unit models – logical combinations of the unit models of their constituent components. Multiple or complex control loops can be easily taken into account by providing their corresponding cause and effect unit models. In particular, the fault tree construction algorithm presented here is based on a component-by-component basis instead of a loop-by-loop or node-by-node basis. The developed tree structure is much more concise and easier to read.

Introduction & Background

Pressure from regulation and society is increasing demands for more rigorous safety and reliability analyses of chemical plants. An essential goal of a Chemical Process Quantitative Risk Analysis (CPQRA) is to estimate the frequency and the consequence of specific incidents. As a powerful tool for risk assessment, Fault Tree Analysis (FTA) has long been successfully applied in CPQRA to predict the likelihood of hazardous accidents and identify major risk contributors.

While well-developed computerized codes are available for the evaluation and analysis of fault trees, the synthesis of fault trees remains the weakest link of the whole procedure. The conventional manual construction of fault trees is a difficult and time-consuming task and susceptible to human errors and omissions. Realizing this problem, many researchers have been focusing on computerized fault tree synthesis for years. Also, some computer-aided fault tree synthesis techniques have been developed [1-6].

Computerized fault tree synthesis is similar to conventional fault tree construction in the sense that it starts with the TOP Event, through all of the identified intermediate gates to the primary failures of the components or boundary conditions. A variety of modeling techniques have been employed to fault propagation in chemical processes. Fussell [1] develop a formal fault tree synthesis methodology for electrical systems. He used failure transfer functions to model failure modes of equipment. In his synthesis tree model, the system-independent component failure transfer functions were used together with the system schematic diagram and associated system boundary conditions to construct the final fault tree. Fussell's algorithm was later modified and improved by Taylor [2]. Taylor used the equation bigraph, the signal flow graph, and the state transition table to model the fault propagation. Lapp and Powers proposed their algorithm based on digraphs [5]. A digraph gives an explicit description of the qualitative relationships between the process variables, human errors, and equipment failures. A. Shafaghi *etal* developed a systematic approach to construct fault trees based on decomposing the plant into a set of control loops [6]. In their method a digraph is created for each control loop and for the plant as a whole. Control loops are treated as the skeleton of the plant, and the final tree structure is more concise. The fault tree synthesis algorithm of B.E. Kelly and F.P. Lees is based on a mini-fault tree model

[3,4]. Their mini-fault trees are generated from propagation equations, event statements, and decision tables.

However, some of the above methods are not applicable to complex systems. Some involve generation of a complex system model, such as digraphs, decision tables, and connection tables between control loops. In particular, control loops are integral parts of a chemical plant. It is crucial in fault tree synthesis since control system is designed to prevent process deviations. Fault tree logics to model the behavior of process loops and control loops are one of the most challenging parts and have been the subject of many literature arguments for years. No entirely-satisfactory algorithm has been published for fault tree synthesis, especially when control loops are encountered.

In this paper, a potential algorithm for computer-aided fault tree synthesis is developed. This methodology starts directly from a process block diagram. Since mini-fault trees avoid the difficult task of generating digraphs and are more adapted to the automatic construction of fault trees, the method presented here is based on a mini-cause and effect tree approach. Mini-fault tree cause and effect unit models are developed for unit operations such as heat exchangers, compressors, pumps, tanks, *etc.* The basic idea is to capture the cause and effect logic between equipment behaviors, human responses, and environment factors around each item of equipment directly into mini-cause and effect trees. Special unit models are developed to treat control loops. Users can tailor the generalized structures of this special unit model, if necessary. Details of the methodology are discussed in the following sections. An example is embedded in the description of the methodology for better understanding. The fault tree generated here is compared to the published result in the analysis section.

Methodology

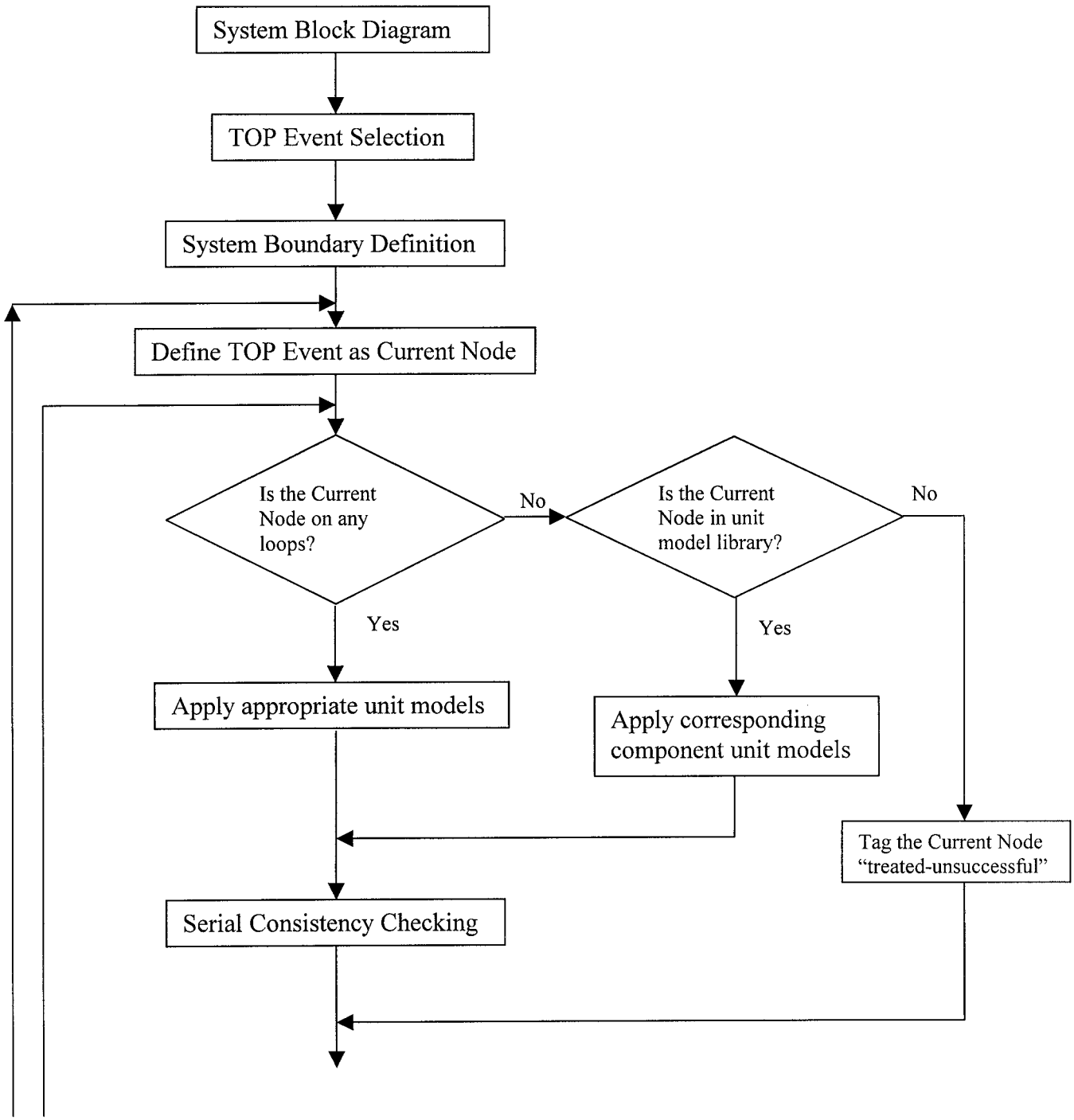
(1). Overview

The first step of FTA is to identify the undesired hazard accidents. This is normally achieved by hazard identification and consequence analysis. In this proposed method, a checklist will be used

to select concerned accidents. Users can also input their perceived incidents into the TOP event library.

In order to computerize the whole procedure, system Pipeline & Instrument Diagram (P&ID) and certain configuration and initiating information must be transformed to computer readable formats. The block diagram is used to represent the system topology. In a system block diagram, a solid line between components indicates the physical connection between them, and a dotted line represents the signal transmission between components (typically in control loops, trip systems, and alarm systems). Configuration information, initiating states, and scope of study are incorporated into the system boundary. Once the TOP Events have been identified, the algorithm proceeds to find the unit models in the TOP Event library to trace the TOP Event down to deviations in the process variables in the library, or prompt for user interaction if it does not exist in the library. Then intermediate events are selected and processed in turn iteratively until system boundary or primary events are reached. The algorithm will detect events not existing in the cause and effect unit model library and prompt for user interaction. After the user responses, the algorithm will resume. Serial consistency is checked during this procedure. Parallel consistency is performed after the initial tree is constructed. After that, a simplification procedure will be performed to make the fault tree more concise.

The following figure is a simplified flow chart of the algorithm. Some steps are described in detail in the following sections. An example will be used to illustrate the working procedure of the algorithm. To simply the problem, environment factors are not considered. To limit the space, failures of all the pipes and flanges are not considered as well. However, it is straightforward to incorporate these factors and failures.


System Block Diagram
TOP Event Selection
System Boundary Definition
Define TOP Event as Current Node
Is the Current Node on any loops?
No
Is the Current Node in unit model library?
No
Yes
Yes
Apply appropriate unit models
Apply corresponding component unit models
Tag the Current Node "treated-unsuccessful"
Serial Consistency Checking

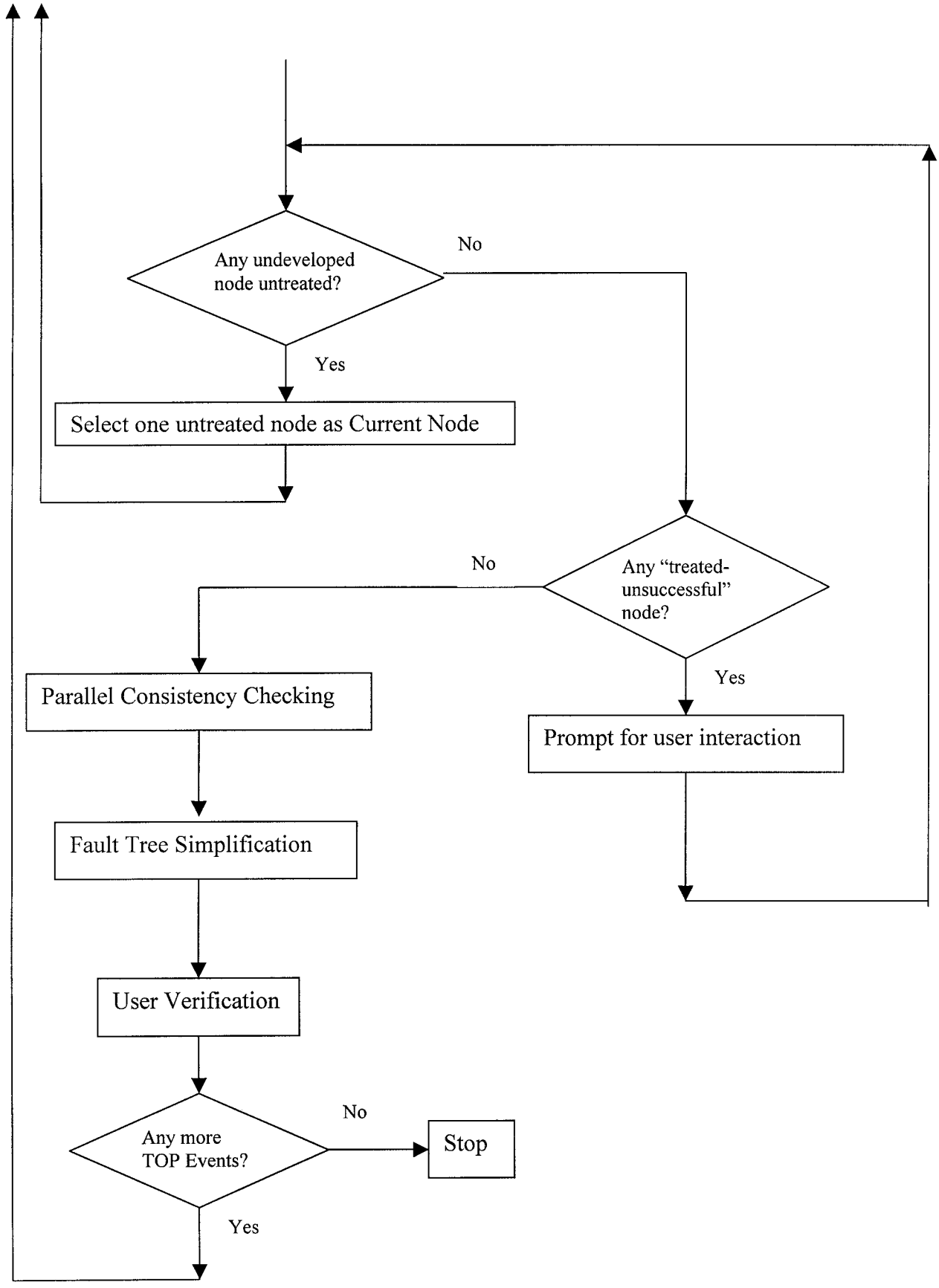


Figure 1 A simplified flow chart of the fault tree synthesis algorithm

(2). System Boundary

Chemical plants are usually very large and complex. It is impractical and unnecessary to analyze the entire plants at one time, but chemical plants can be easily divided into functional units. To gain a reasonable and sufficient scope of study, the first task is to divide the plant into a set of units and then examine the specified units one by one. Therefore, before applying the methodology, the concerned system and its physical boundary must be well defined. System boundary defines the initial states of components and the environment factors to be considered as well.

(3). System Block Diagram

In order to automate fault tree synthesis, an appropriate modeling technique must be specified. Before applying the modeling technique, the computer should be able to interpret the system P&ID and other necessary information. In this algorithm, a system block diagram is used to represent the system topology. In a system block diagram, a solid line between components indicates the physical connection between them, and a dotted line represents the information transmission between components (typically in control loops, trip systems, and alarm systems). Each block in a system block diagram represents one item of equipment as illustrated by the simple example presented in Figure 2, which is adapted from [5]. This process is to cool a hot nitric acid stream before entering the reactor where it reacts with benzene to produce nitrobenzene.

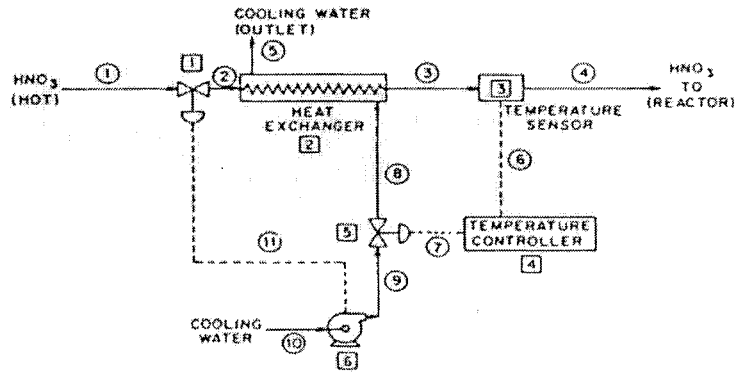


Figure 2. Nitric acid cooler with temperature feedback and pump-shutdown feedforward [5]

The corresponding block diagram is shown below in Figure 3. The enclosed dotted rectangle represents the physical boundary of our study. The trip valve and the control valve here are both air-to-open which will close upon loss of instrument air. The signals of temperature sensor and controller will increase with increasing inputs.

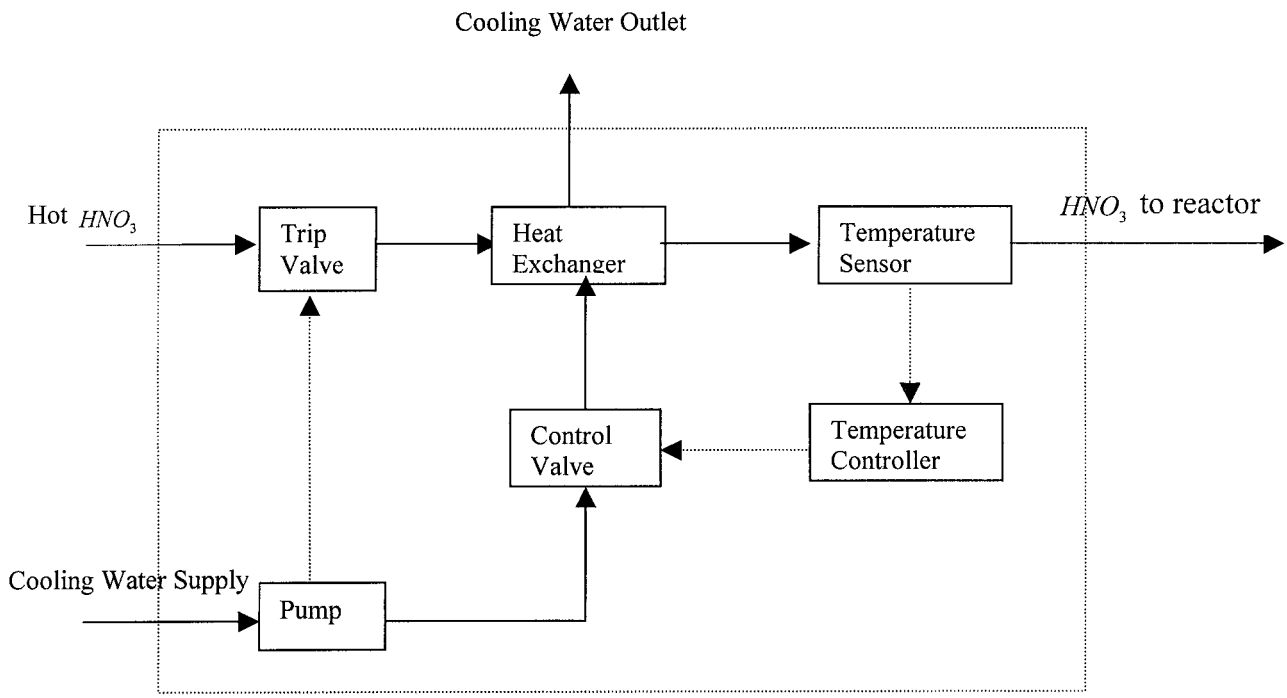


Fig. 3. System block diagram for the above nitric acid cooling process

(4). TOP Event (Accident) library

TOP Event is defined as the undesirable event or incident. The identification of TOP Events is critical but often underestimated. Undesirable accidents can be categorized into three classes as human impacts, environmental impacts, and economic impacts [7]. Typical TOP Events include release of toxic materials, fires, explosions, human injury, environment contamination, property damage, poor product yield/quality, legal liability, *etc.* Each of these accidents can be subdivided by their characters and consequences. For instance, a fire can be a flash fire, pool fire, jet fire, or a BLEVE. Industry has used checklists, safety audit, What-if Analysis, HAZOP, and Cause and Consequence Analysis to identify major hazard accidents [7]. It is rather difficult to automate TOP Event identification. In this paper, a checklist will be used for the algorithm to identify

undesired events automatically. A checklist is a list of questions about material information, process operations and plant management. The algorithm will then generate a list of the "possible" major hazard accidents based on the answers to the checklist. Users can then select which events to examine further from the list. Users can also input identified TOP Events directly.

TOP Events are normally undesired accidents such as fires or explosions. Therefore they must be traced to some of process deviations to start the automatic fault tree construction. At this moment, user interaction is required to do this. However, this process can be partially automated by providing special cause and effect unit models for typical TOP Events.

For the above nitric acid cooler, one of the identified accidents can be that runaway reaction occurs in the benzene reactor. Many factors such as large external fire outside the reactor and feeding too fast can cause a runaway reaction. In our system boundary, the only cause of this accident is hot HNO_3 to the reactor. The cause and effect unit model for this TOP Event is:

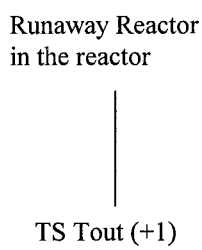


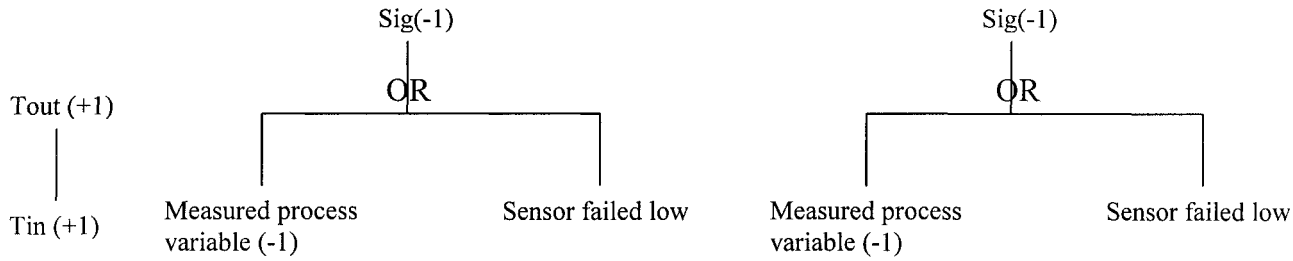
Figure 4. "Runaway Reactor in the reactor" cause and effect model for the nitric acid cooler

(5). Cause and Effect Unit Model Library

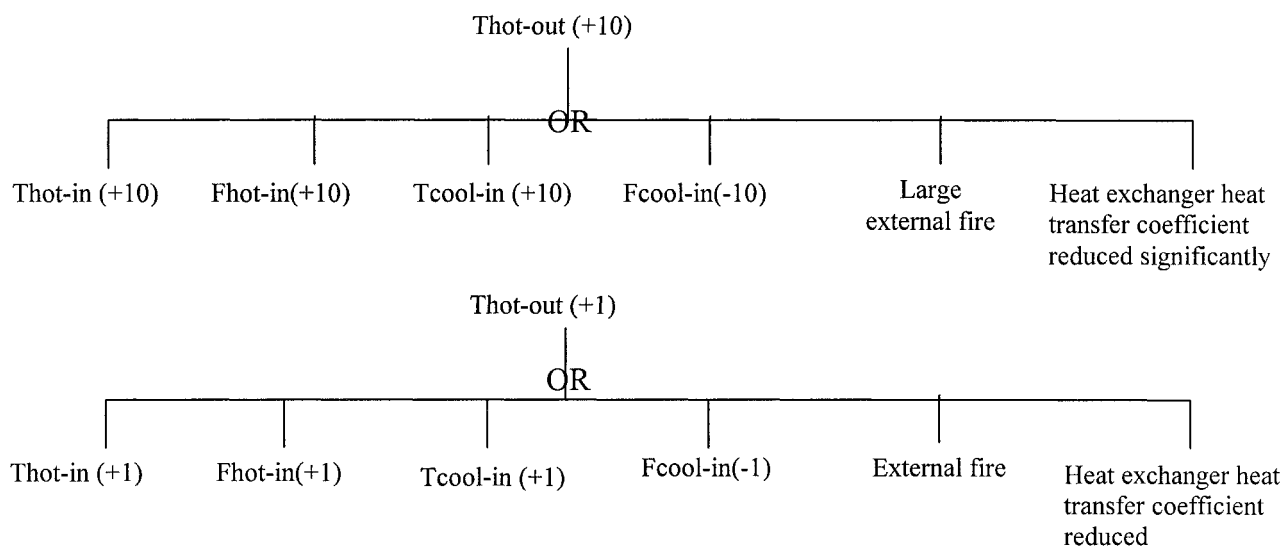
Each chemical plant is unique, however, they have much identical equipment, materials, and even processes. For these common components, we have sufficient knowledge of their failure modes. If we can utilize this knowledge and automate this part, it will reduce the labor needed significantly. There is one cause and effect unit model in the unit model library corresponding to each item of equipment. Each cause and effect unit model may consist of several mini-cause and effect trees depending on the specific component. When constructing the fault tree, the algorithm will search for appropriate mini-fault trees in the model and use them.

To save space, we list some cause and effect unit models to be used later in the nitric acid cooler example. These figures are only for illustration purpose. The actual library contains more mini-cause and effect trees, and they may be more complex. In particular, the heat exchanger in the above system is special in that there will be an exothermic reaction if there is an internal leakage. This failure mode is neglected here to simply the problem.

Temperature Senor:

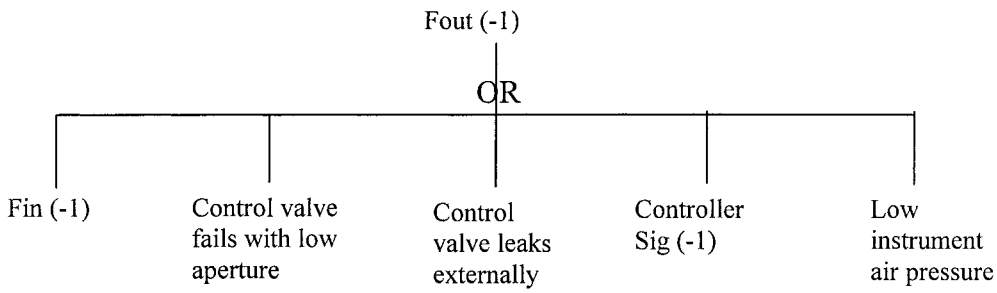


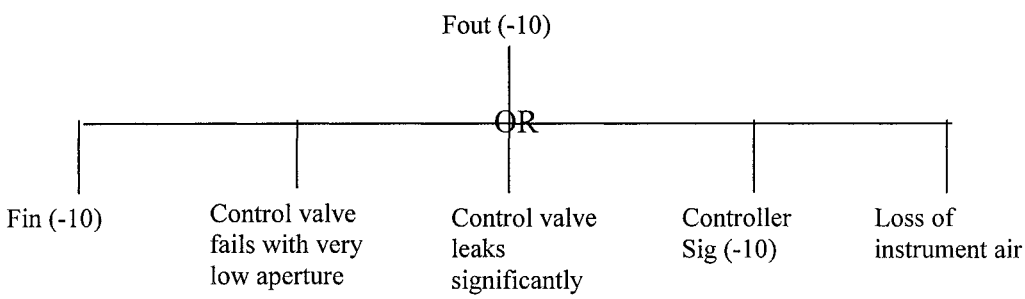
Heat Exchanger:



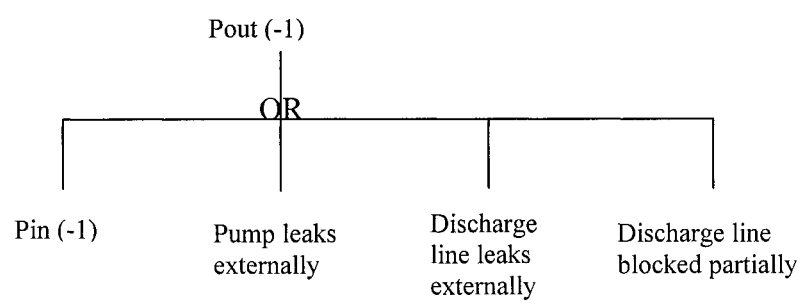
Trip Valve:

Control Valve (Air to Open):





Pump:



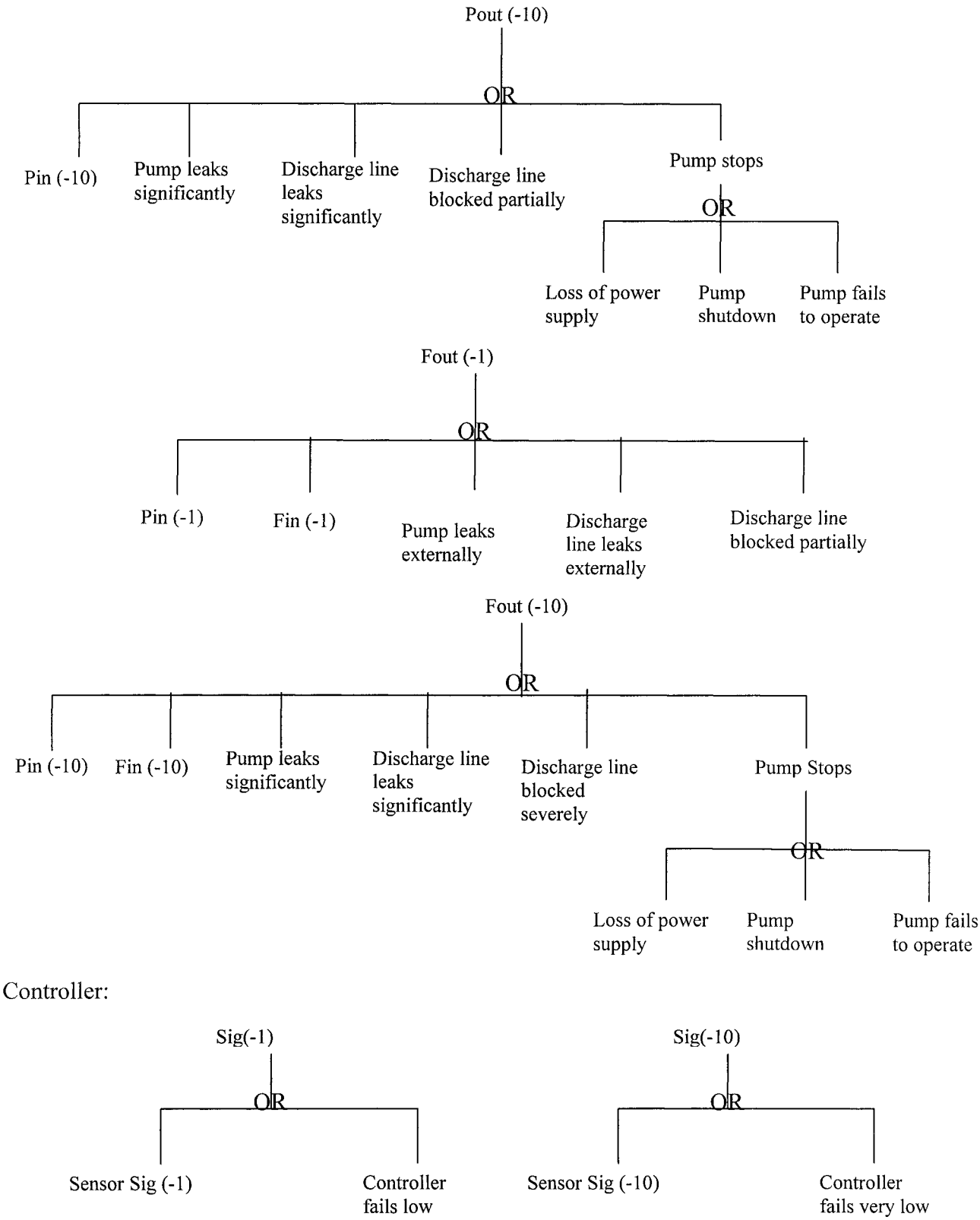


Figure 5. Some unit models for the nitric acid cooling process

(6) Cause and Effect Models for Control loops

Control loops are integral parts of a chemical plant. It is crucial in fault tree synthesis since control system is designed to prevent process deviations. The cause and effect unit models for control loops are basically logical combinations of the unit models of their constituent components. Different types of control systems have different unit models. It is not easy to identify and classify control loops automatically. In this algorithm, user interaction is required to input certain information such as control types and controlled variables for control systems.

There are three circumstances that a controlled process variable can deviate beyond its normal range [5]. Firstly, uncontrollable disturbances can drive the controlled variable to abnormal states. Secondly, a deviation can be caused by a controllable disturbance while the control loop is inactive. Finally, sometimes the control loop itself can cause process deviations. Based on this, a generalized cause and effect unit model is proposed for control loops:

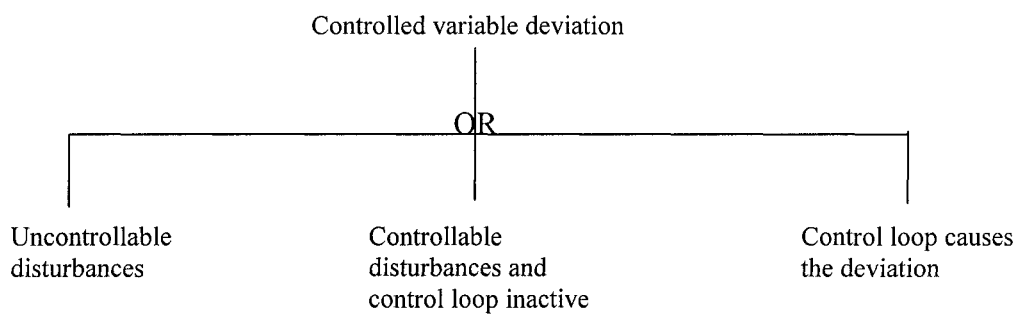


Figure 6. The unit model for control systems

Different types of control systems have different mechanisms of failures. Feedback control, feedforward control, nested feedback control, and flow ratio control loops are typically used in chemical processes. Among them, negative feedback control and negative feedforward control are the most popular control schemes. Computer control, alarm system, and manual control systems can be viewed as special forms of feedback or feedforward systems. In this algorithm, each type of control loops has its corresponding cause and effect model, which has three mini-cause and effect trees for "Uncontrollable disturbances", "Controllable disturbances and control loop inactive", and "Control loop causes the deviation" events. In the nitric acid cooling process, the temperature control system is a typical negative feedback control loop. The pump shutdown

trip system is a negative feedforward control loop. However, upon loss of instrument air, both the control valve and trip valve will close. Therefore "loss of instrument air" will cause shutdown of the nitric acid cooling process. This is viewed as a negative feedforward control loop. Figure 7 and Figure 8 display the cause and effect unit models for the negative feedback control loop and the feedforward control loop. User interaction is required for the cause and effect model of "control loop causes the deviation" event in feedforward control loops.

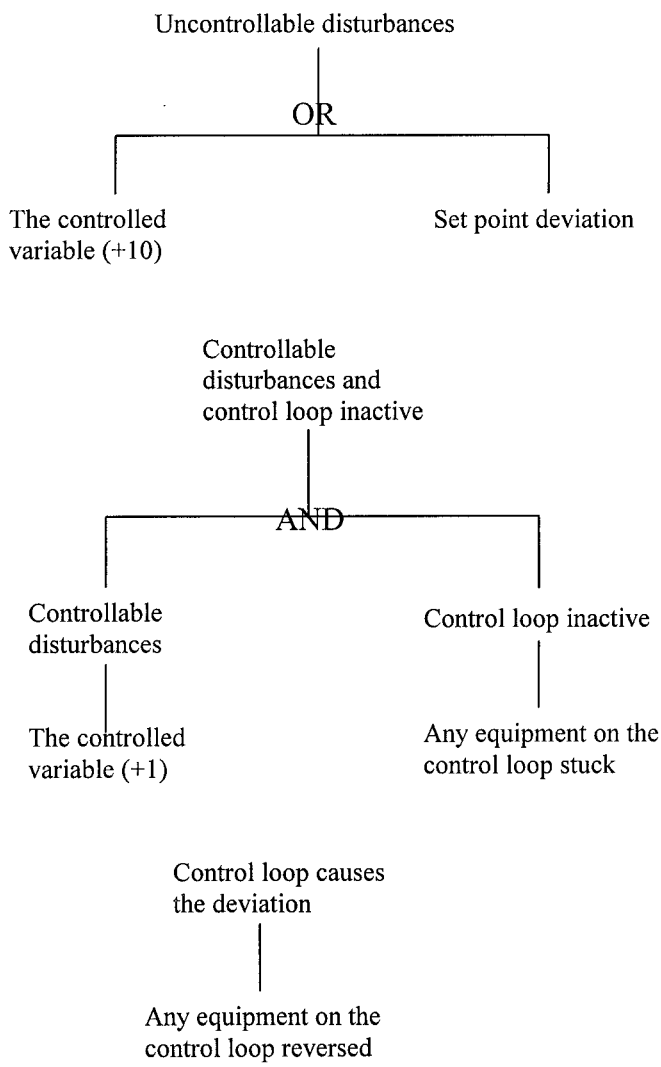


Figure 7. The unit model for negative feedback control loops

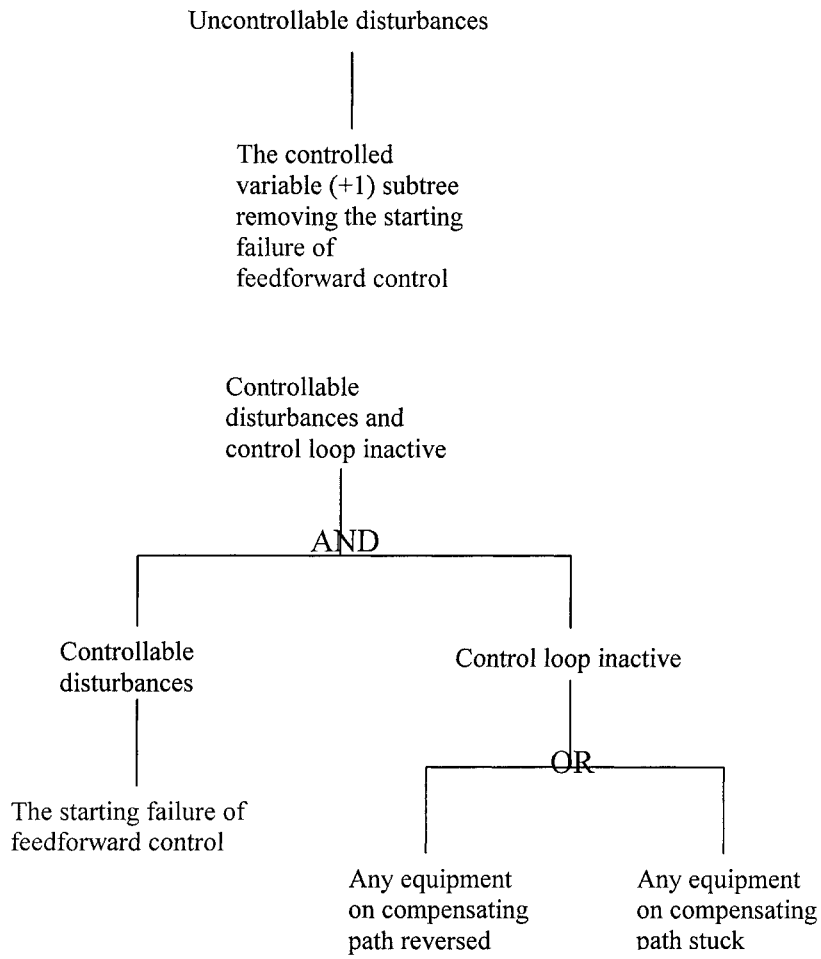


Figure 8. The unit model for negative feedforward control loops

(7). Consistency Checking

It is essential to check the consistency among tree events to remove inconsistent and unreasonable events. There are two types of consistency – serial and parallel consistency. Serial consistency is the consistency of events within its upper level events. Parallel consistency is the consistency of events between two branches of the same AND gate. During the fault tree construction, the program will store all the upper level parents, check for serial consistency, and delete the inconsistent events whenever encountered. However, parallel consistency checking cannot be performed during construction. It is also necessary to ensure that events in the fault tree does not violate or go beyond the system boundary. This step can be done during fault tree construction.

(8). Initial Fault Tree Synthesis

As stated in part (4), one of the identified accidents can be runaway reaction in the reactor in the nitric acid cooling process. Within our system boundary, the only cause is hot HNO_3 to the reactor. According to the system block diagram, this is equivalent to "TS Tout (+1)". Applying the cause and effect model of temperature sensor, "TS Tout(+1)" can be caused by "TS Tin(+1)". The temperature sensor is connected to the heat exchanger in the block diagram, so this is trace to "HE Thot-out(+1)".

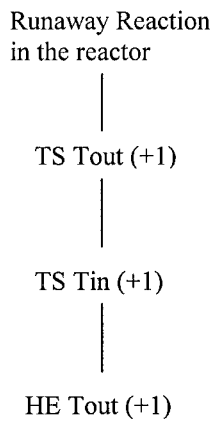


Figure 9. The subtree for the TOP Event "runaway reactor in the reactor"

HE Tout (+1) is the controlled variable of the temperature control system – a negative feedback control loop. Applying the generalized unit model for the control loops, we obtain:

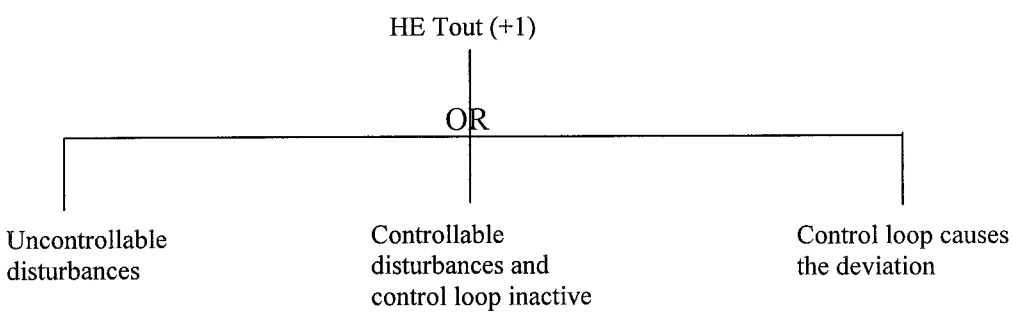


Figure 10. The generalized unit model for event "HE Tout (+1)

After applying the cause and effect unit model of negative control loops, the unit models for "Thot-out (+10)" and "Thot-out (+1)" of the heat exchanger are used separately. Thus the fault

tree is traced to the output variable deviations of the trip valve, control valve, and pump subsequently. This procedure continues until the system boundary or primary failures are met.

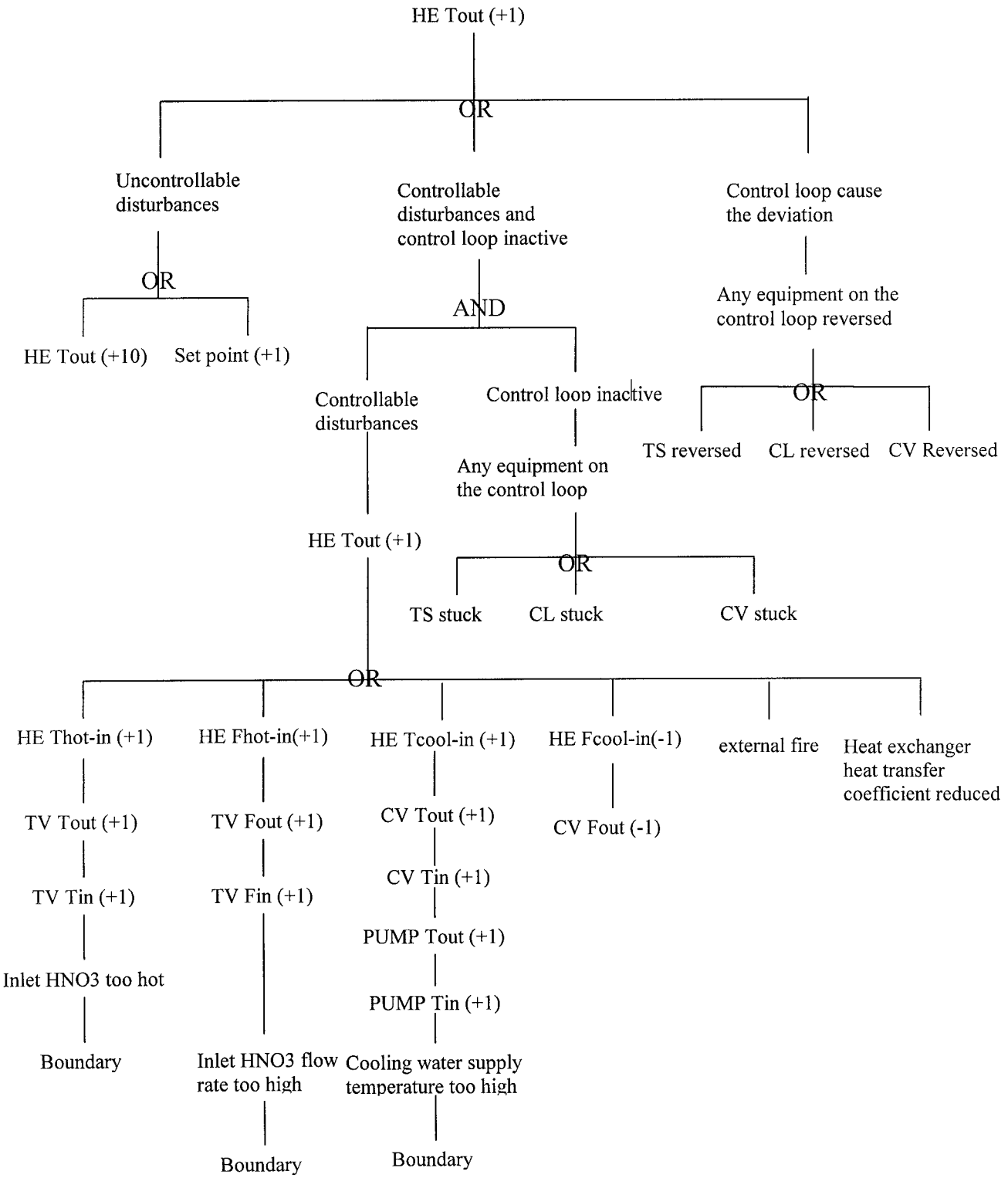


Figure 11. The subtree for HE Tout (+1)

When intermediate event " TS Sig (-1)" is traced, one of its child events is "HE Thot-out (-1)". Serial consistency checking reports that this event is inconsistent with its parent node " HE Thot-out (+1)". Therefore this event is deleted as shown in the following subtree.

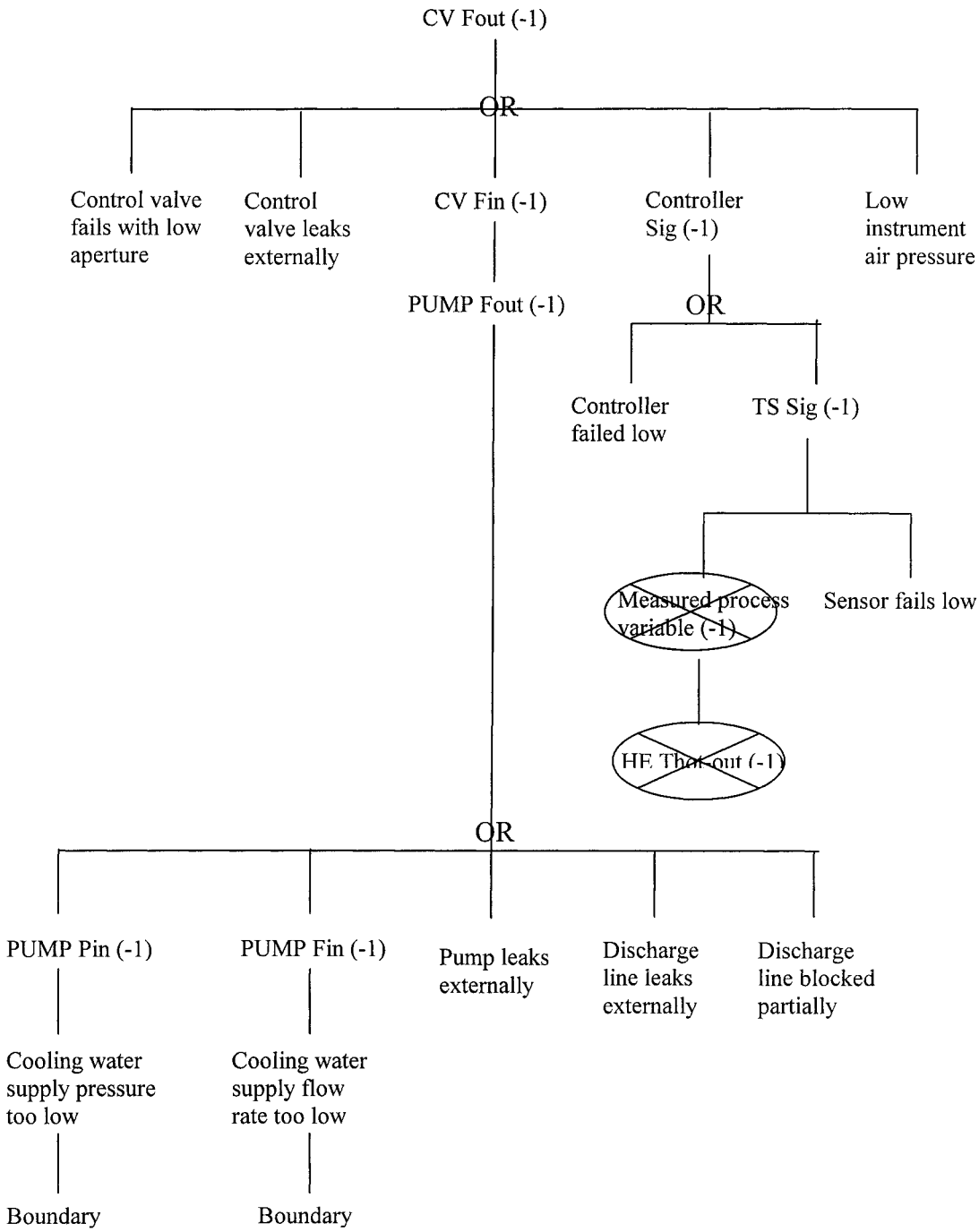


Figure 12. The subtree for CV Fout (-1)

Following a similar deductive way, we obtain the subtree for the intermediate event "HE Thot-out (+10)" in Page 19. As in the previous figure, "TS Sig (-10)" only has one input – "sensor fails very low" because of serial inconsistency.

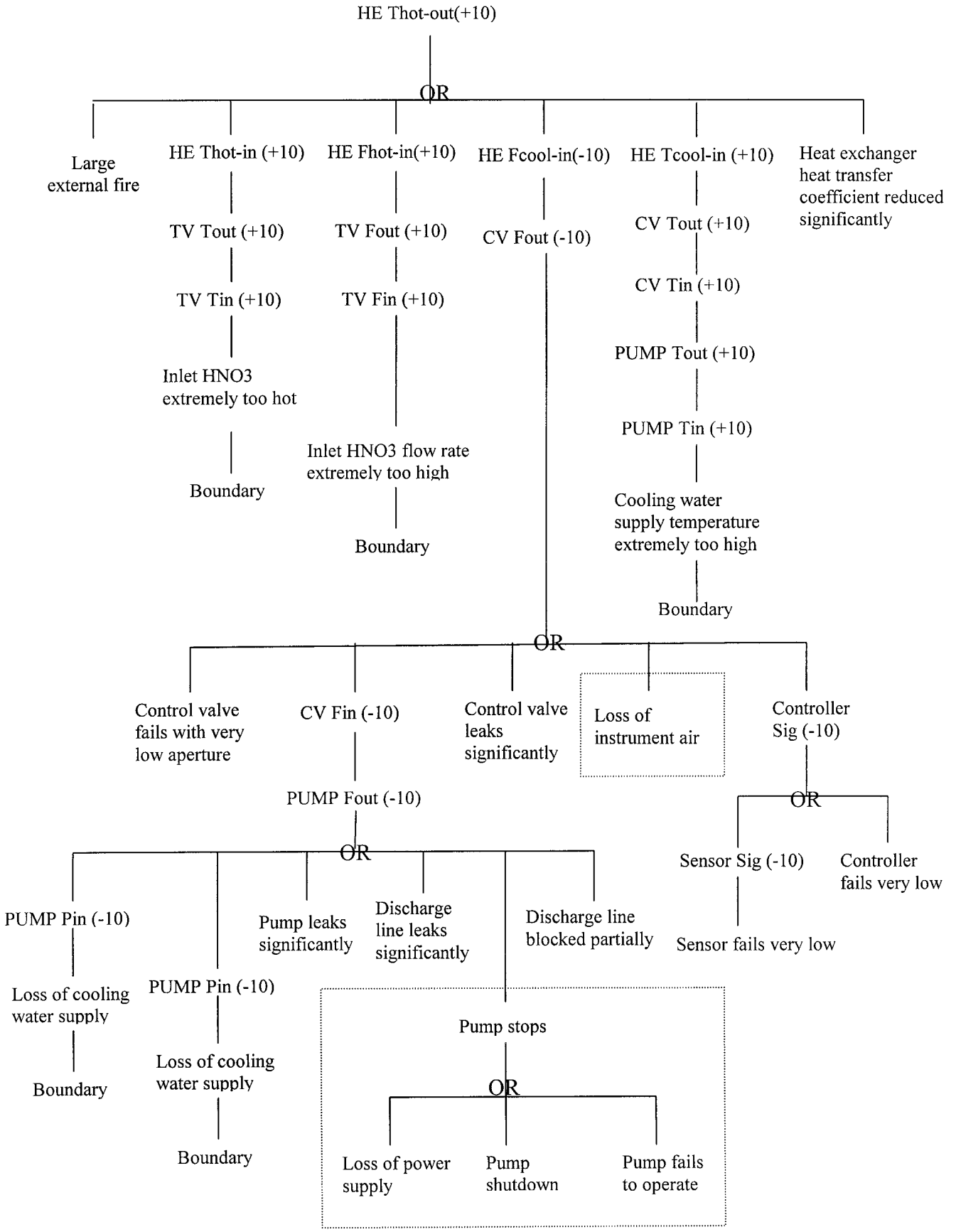


Figure 13. The subtree for HE Thot-out (+1)

If there is no emergency shutdown system, the above fault tree is correct. However, "Pump stops" is the starting point of the trip system (negative feedforward control). The compensating path for this "Pump stops" event is the signal line and trip valve. Therefore the "Pump stops" event must be replaced by the "Pump stops and the trip system inactive" gate below to take the negative feedforward system into account. The process will also be shutdown by "Loss of instrument air". "Loss of instrument air" is viewed as the starting deviation of a feedforward control loop as well. The corresponding compensation is the closure of the trip valve upon loss of instrument air. Therefore, in the fault tree, these two events are replaced by the unit model of negative feedforward control loops.

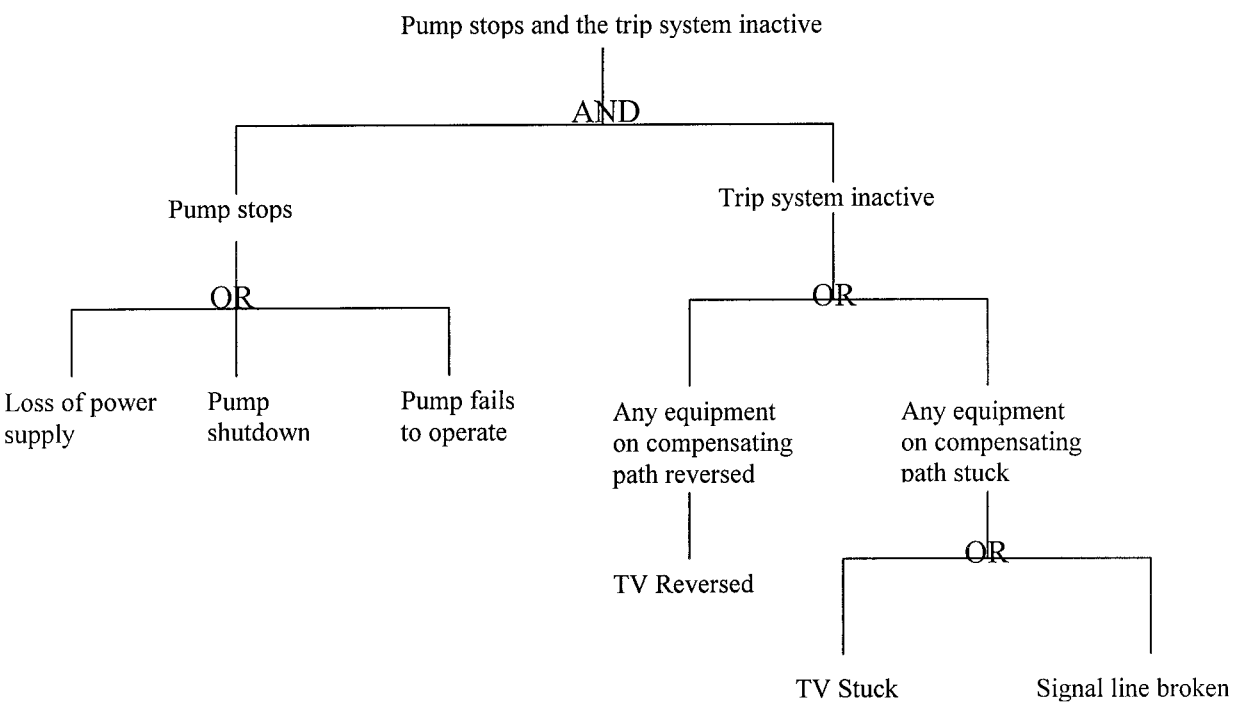


Figure 14. The subtree for "pump stops and the trip system inactive"

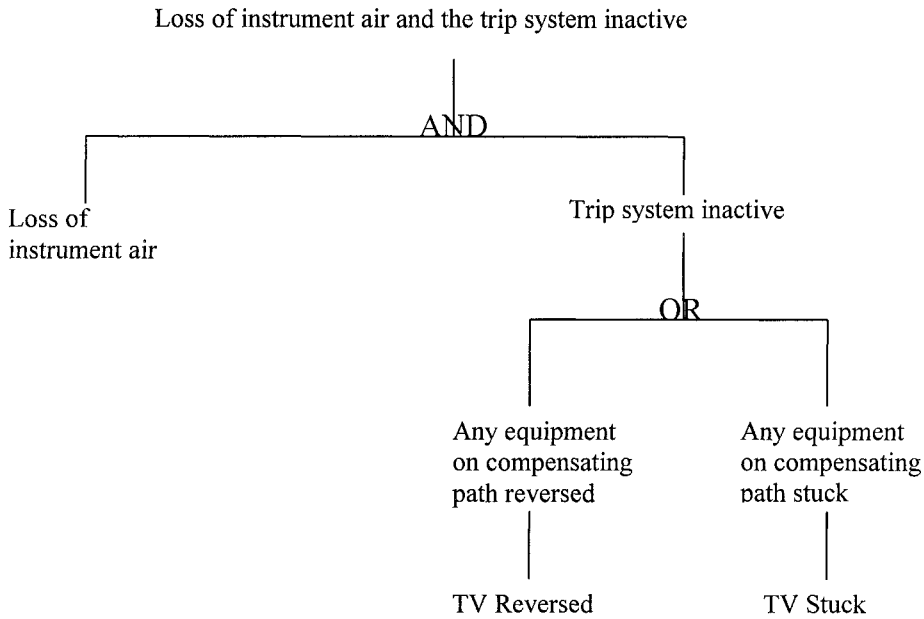


Figure 15. The subtree for "loss of instrument air and the trip system inactive"

After the initial fault tree have been developed, the algorithm will run parallel consistency checking to remove any inconsistent events between the two arms of AND gates. This example does not have parallel inconsistency.

(9). Simplification

Fault trees drawn directly from a computer algorithm are normally opaque. A simplification procedure can make them concise and understandable to humans. Two kind of simplification will be applied – algebraic simplification and tree simplification. When a certain event (with the probability of 1) is under an OR gate, algebraic simplification is performed to remove the parent gates until an AND gate is encountered. If an impossible event (with the probability of 0) is under an AND gate, removal will continue until an OR gate is met. Tree simplification mainly refers to tree suppression. All intermediate events with only one child are removed during tree suppression. After algebraic simplification and tree simplification, identical events under a gate are identified and removed automatically.

From part (8), we already have the initial fault tree for the runaway reaction top event. Figure 16 is the final tree after simplification. Gates that are not noted as AND are OR gates.

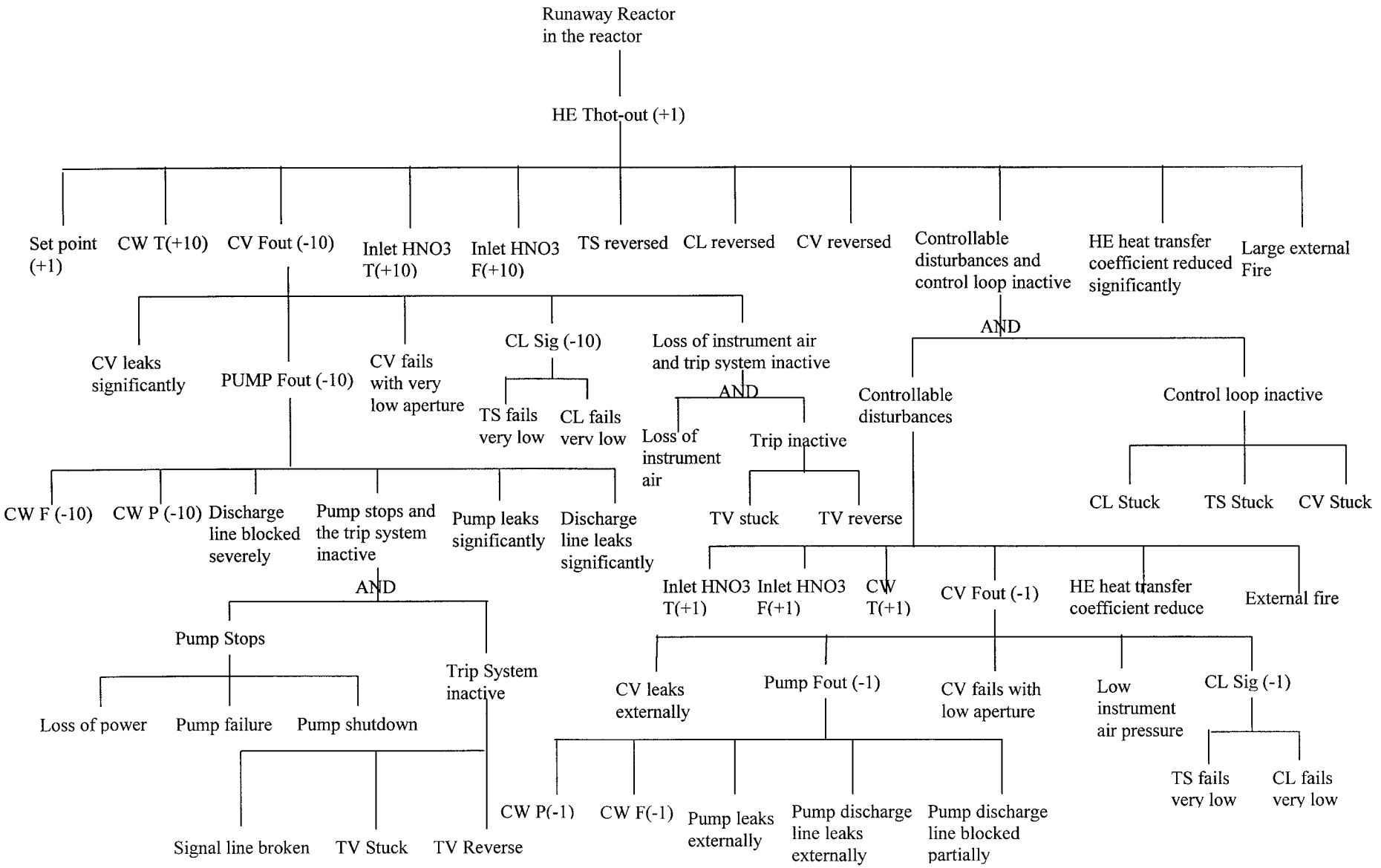


Figure 16. The final fault tree for the nitric acid cooler

(10). User Interaction

Though a computer will take out the dull part of the methodology, human-machine interfaces are required in many places to select TOP Events, input loop configuration information, and incorporate the cause and effect unit models for special process components, *etc.* Users are responsible for verifying the final fault tree structure. User decision will override a computer most of the time during or after the tree construction and consistency checking stages.

Analysis

Many failures in the fault tree in Page 24 are not considered in the published fault tree in [5]. In order to compare these two fault trees, the analysis resolution should be consistent. By assigning those failures that are not considered in Lapp and Powers' paper a probability of 0 and remove them from OR gates, we can obtain the following simplified fault tree. Gates that are not noted as AND are OR gates.

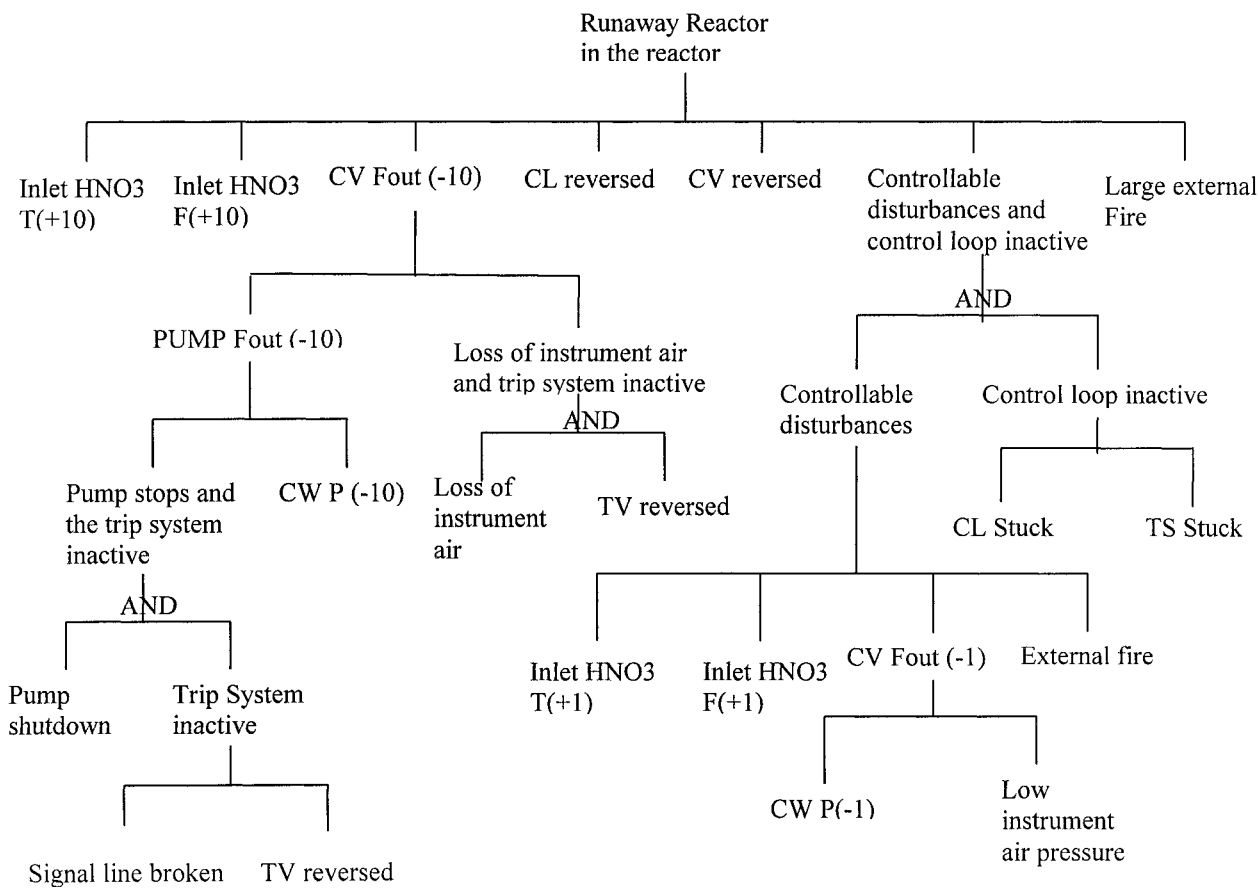


Figure 17 The simplified fault tree for nitric acid cooler

After algebraic analysis, Minimum Cut Sets (MCS) of this simplified fault tree are:

{Inlet HNO_3 T (+10)}

{Inlet HNO_3 F (+10)}

{Large external fire}

{CL reversed}

{CV reversed}

{CW P(-10)}

{Loss of instrument air, TV reversed}

{Pump shutdown, TV reversed}

{Pump shutdown, Signal line broken}

{Low instrument air pressure, CL stuck}

{Low instrument air pressure, TS stuck}

{Inlet HNO_3 T(+1), CL stuck}

{Inlet HNO_3 T(+1), TS stuck}

{Inlet HNO_3 F(+1), CL stuck}

{Inlet HNO_3 F(+1), TS stuck}

{External fire, CL stuck}

{External fire, TS stuck}

{CW P(-1), CL stuck}

{CW P(-1), TS stuck}

As pointed by [8], the probability of the event "Low air pressure of the cooling water control valve" and the event " control valve reversed" occurring simultaneously is negligible. We can simply replace the EOR (exclusive or) gate as OR gate. The MCS for the published fault tree in [5] are:

{Inlet HNO_3 T (+10)}

{Inlet HNO_3 P (+10)}

{Large external fire}

{CL reversed}

{CV reversed}

{CW P(-10)}

{Loss of instrument air}

{Pump shutdown, TV reversed}

{Pump shutdown, Signal line plugged}

{Low air pressure (controller), CL stuck}

{Low air pressure (controller), TS stuck}

{Inlet HNO_3 T(+1), CL stuck}

{Inlet HNO_3 T(+1), TS stuck}

{Inlet HNO_3 P(+1), CL stuck}

{Inlet HNO_3 P(+1), TS stuck}

{External fire, CL stuck}

{External fire, TS stuck}

{CW P(-1), CL stuck}

{CW P(-1), TS stuck}

The minimum cut set {Loss of instrument air, TV reversed} is different from that in the published paper {Loss of instrument air} because Lapp and Powers did not consider the fact that the trip valve will close upon loss of instrument air. Except for this, the two minimum cut sets are consistent.

Conclusion

A new potential methodology to construct fault trees automatically is proposed in this paper. System block diagram and cause and effect unit models are employed to model chemical processes, and a simple example is shown. From the analysis section, we can see the fault tree developed by this new algorithm is equivalent to the published result. This algorithm is advantageous in many aspects. The algorithm works directly from the system block diagram and avoids the tedious working of drawing digraphs, transition tables, *etc.* Control loops are considered and treated by special cause and effect unit models – logical combinations of the unit models of their constituent components. Multiple or complex control loops can be taken into account by providing the cause and effect unit models. In particular, the fault tree construction

algorithm presented here is based on a component-by-component basis instead of a loop-by-loop or node-by-node basis. The tree structure developed is much more concise and easier to read.

To be programmed in computer codes and applied in a real case, the algorithm described above must be tested against substantial examples. Any computer codes must be examined carefully and extensively verified before used in a real application. It is not recommended to use the computer-aided approach alone, since many of the problems in the system design can be discovered during study of the process. However, the computer-aided fault tree synthesis can be an initial stage of FTA or as an independent check or to supplement a manual analysis.

Acknowledgement

This project is under the support of Mary Kay O'Connor Process Safety Center, Chemical Engineering Department, Texas A&M University.

Notation

T	Temperature
P	Pressure
F	Flow rate
out	Outlet
in	Inlet
Sig	Signal
TS	Temperature Sensor
TV	Trip Valve
CV	Control Valve
HE	Heat Exchanger
CL	Controller
CW	Cooling Water Supply
hot-out	Outlet of the hot fluid in a heat exchanger
hot-in	Inlet of the hot fluid in a heat exchanger
cool-out	Outlet of the cooling fluid in a heat exchanger
cool-in	Inlet of the hot fluid in a heat exchanger
+1	Median increase
+10	Large increase
-1	Median decrease
-10	Large decrease
AND	And gate
OR	Or gate

References

1. Fussell J.B., A Formal Methodology for Fault Tree Construction, Nuclear Science & Engineering 52 (1973), 421-432
2. Taylor, J.R., An Algorithm or Fault Tree Construction, *IEEE Transactions On Reliability*, R-31 (1982), 137-146
3. Kelly B.E., F.P. Lees, The Propagation of Faults in Process Plants: 1. Modeling of Fault Propagation, *Reliability Engineering* 16 (1986), 3-38
4. Kelly B.E., F.P. Lees, The Propagation of Faults in Process Plants: 2. Fault Tree Synthesis, *Reliability Engineering* 16 (1986), 39-62
5. Lapp S.A., G.J. Powers, Computer-aided Synthesis of Fault-trees, *IEEE Transactions On Reliability* R-26 (1977), 2-12
6. Shafaghi, A. F.P. Lees, P.K. Andow, An Illustrative Example of Fault Tree Synthesis Based on Control Loop Structure, *Reliability Engineering* 8 (1984), 193-233
7. Center for Chemical Process Safety AICHE, Guidelines for Hazard Evaluation Procedures – Second Edition with Worked Examples, American Institute for Chemical Engineers, 1992
8. Lapp S.A., G.J. Powers, Update of Lapp-Powers Fault-tree synthesis Algorithm, *IEEE Transactions On Reliability* R-28 (1979), 12-15