COMPUTER ASSISTED FAULT TREE SYNTHESIS

Submitted In Partial Fulfillment Of The

Requirements For The Degree Of

Doctor Of Philosophy

By Steven Anthony Lapp

Department Of Chemical Engineering

Carnegie-Mellon University

Pittsburgh, Pennsylvania 15213

January 12, 1978

COMPUTER ASSISTED FAULT TREE SYNTHESIS

Submitted In Partial Fulfillment Of The

Requirements For The Degree Of

Doctor Of Philosophy

By Steven Anthony Lapp

Department Of Chemical Engineering

Carnegie-Mellon University

Pittsburgh, Pennsylvania 15213

January 12, 1978

Carnegie-Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE Computer Assisted Fault Tree Synthesis

PRESENTED BY Steven Anthony Lapp

ACCEPTED BY THE DEPARTMENT OF Chemical Engineering

ABSTRACT

Fault tree analysis is a quantitative system safety technique for identifying the chains of events which could cause a specific hazard. The formal strategy presented in this thesis increases the accuracy and speed by which fault trees may be generated and allows fault tree analysis to be used by a wider range of practicing engineers. Digraph models are used to describe the normal, failed, and conditional relationships for individual process components. Component models are linked to form a model for the system under study. An algorithm is presented which directly deduces the fault tree from the system digraph. Methods for modelling human interactions and analyzing sequential systems are also presented.

ACKNOWLEDGEMENTS

I have enjoyed performing the work described in this thesis. The research has been challenging and satisfying. I wish to thank Dr. Gary Powers for his valuable guidance. His encouragement and many suggestions added much to the quality of this work. Special appreciation is also due to those members of the Advanced Process Synthesis class who applied this technique to real systems. Valuable information concerning the method was obtained from these studies.

I also wish to thank Rockwell International Corporation for the fellowship which supported me at Carnegie-Mellon.

Finally, I want to express my appreciation to my wife Dana. Her love and continual encouragement made the road a lot easier to travel. I find it difficult to imagine anything without her.

TABLE OF CONTENTS

INTRODUCTION

Fault Tree Analysis was originally developed by Bell Telephone Laboratories in 1961. Bell used the technique to evaluate the launch control system of the Minuteman Missle [1]. In applying the Fault Tree Method, one begins by identifying some undesirable event associated with the system. This event is termed the Top Event. For the Minuteman case, examples of this might be (1) Inadvertent firing of the missle or (2) Failure to launch when called upon to do so. Once an undesirable event has been chosen, the analysis proceeds by asking "What could cause this?" In answering this question,one generates other events connected by the logical operators AND, OR, and EOR. EOR is an abbreviation for "Exclusive Or". This operator has two inputs and will produce a TRUE output if one, but not both, of them is TRUE. Successive events are then developed in a similar manner. The analysis terminates when events are encountered which cannot be developed further. These events are called primal events. The logical structure relating the Top Event to the primals is the Fault Tree.

Once constructed, the Fault Tree can be of considerable value in determining the paths whereby primal events may propagate through the system to cause the Top Event. Algorithms currently exist [2] that determine which primal events, or combinations of primal events, will cause the Top Event for a given tree. These sets of events are termed cut sets. If occurrence rate data are available for the primal

events, a number of useful statistics can be computed [3]. Among these are the probability and rate of occurrence of the Top Event, the probability that a given cut set will occur, and the relative importance of each primal event to the Top Event. Such statistics can provide valuable information as to which part of the system should be modified to decrease the probability of the Top Event. An excellent example of the use of the Fault Tree technique and resulting statistical analysis can be found in the Rasmussen Report on Nuclear Reactor Safety [4].

While much of the statistical and cut set analysis has been automated, actual construction of the Fault Tree is usually done by hand. Manual construction of the Tree can be extremely time consuming (25 man-years of effort was required in the Rasmussen Study [4]). In addition to the time involved, the possibility exists that different analysts may produce different trees [5] either by incorrect logic or omission of certain events. A computer aided synthesis technique would prove valuable in alleviating both of the above problems. Any system of constructing Fault Trees should have the following characteristics:

(1) It should be able to handle complex systems in an efficient manner.

(2) In constructing the tree, system topology as well as actual components should be considered. The topology includes the system surroundings as well as operating and maintenanace procedures. If

the topology is time dependent, this should also be considered.

(3) It should handle multivalued logic ie. the direction and magnitude of deviations in process variables should be considered in addition to component failures. The range of allowed deviations should be greater than Ø,1.

(4) During tree construction, checks should be made to ensure consistency among events. For example, an increase in the temperature of some stream cannot be caused by a simultaneous decrease in the same stream's temperature (More on this in the next section).

Although other criteria certainly exist, these four represent basic requirements for any synthesis system. Table 1 summarizes previous and current work in terms of these criteria.

The issue of a formal synthesis method has been addressed by Fussell [6]. His technique, known as the Synthetic Tree Model, uses mini fault trees as models for component failures to construct the final tree. The system under analysis is segmented into various parts or coalitions and it is through this grouping that consistency requirements arise. The intermediate event under consideration along with any consistency requirements determine which mini fault tree is used.

Taylor's method [7] uses algebraic models for

	Complex Systems	Topology Considered	Multivalued Logic	Consistency Checks	Computer Program	Analyses Performed	Largest Problem (No. Gates in tree)
Fussell (Fault Tree Analysis)	No	Partially (Component Coalitions)	No	Yes (Not Allowed Boundary Conditions)	Yes (Certain Types of Electrical Systems)	<5?	22
Taylor (Cause-Consequence Analysis)	No	Yes	Yes	Yes	No	<5?	Not Applicable (Equivalent Tree would contain about 15 gates.)
Powers & Tompkins (FTA)	No	Partially	No	No	No	5	15
Salem (FTA)	No	Yes	No	Yes	Yes	<5?	40
Lapp & Powers (FTA)	Yes	Yes	Yes	Yes	Yes	30	224

Table 1: Summary of Fault Tree Synthesis Work

components with qualifiers to indicate which equation(s) describe the operation or failure of the component. These qualified equations are then written for each component and the resulting collection forms the system model. This model may then be used to determine the consequences of any deviation in the input variables. The method involves more Cause-Consequence than Fault Tree Analysis.

Tompkins and Powers [8] have suggested a method using input-output models for equipment. These models convey information regarding variable relationships when the components are working as well as the effects of various component failures. Construction of the Fault Tree begins with the identification of important deviations in process variables. The process is then searched for sources of these deviations and it is through this search that the tree is built.

Salem [9] has developed a method based on the use of decision tables. The decision tables form the basic component models from which the fault tree is built. Fault tree construction begins with a definition on the top event which is then developed using decision tables which contain it. Intermediate events are expanded in a similar manner until only primal events are undeveloped in the tree.

The methods discussed so far have dealt with organizing construction of the Fault Tree. Actual use of these techniques still requires a large amount of time, hence the need for some type of computer assistance is still evident. Fussell has programmed his method but the program apparently handles only certain types of electrical circuits [6].

Salem's method has also been programmed, however, events which are normally true are developed in addition to abnormal events [9]. This incurs significant overhead when synthesizing the tree. The other two techniques have not been programmed.

In order to develop a program which performs the synthesis, it is first necessary to develop some means of system modeling which is suited for computer processing. This representation must also be general so that any type of process may be analyzed. With the system thus modeled, the next step is to develop an algorithm for Fault Tree Synthesis using it. These problems of suitable representation and subsequent Fault Tree generation form the core of this thesis.

FAULT TREE SYNTHESIS - A PROTOCOL

As an example of how one might manually generate fault trees, consider the flowsheet shown in Figure 1. The function of this process is to cool a hot HNO3 stream before reacting it with Benzene to form Nitrobenzene. One top event for the system is a high temperature in the HNO3 reactor feed since this could cause a reactor runaway. Consider constructing a fault tree for this event. The following notation is used to describe deviations in process variables. "T", "P", and "M" denote deviations in temperature, pressure, and mass flow respectively. "+" and "-" denote directions of the deviation (positive or negative). "Ø","1", and "1Ø" denote magnitudes of the deviation (none, moderate, or very large). Hence, to represent a large decrease in the mass flow rate of stream 8, write M8(-1Ø).

An engineer would begin by asking "What could cause T4(+1)?" Since stream 4 is directly connected to stream 3 through the temperature sensor, he might reason that T3(+1) is the cause so the first step in constructing the tree would be to place an OR gate under T4(+1) with T3(+1) as its input ie.

T4(+1)
*
OR
*
T3(+1)

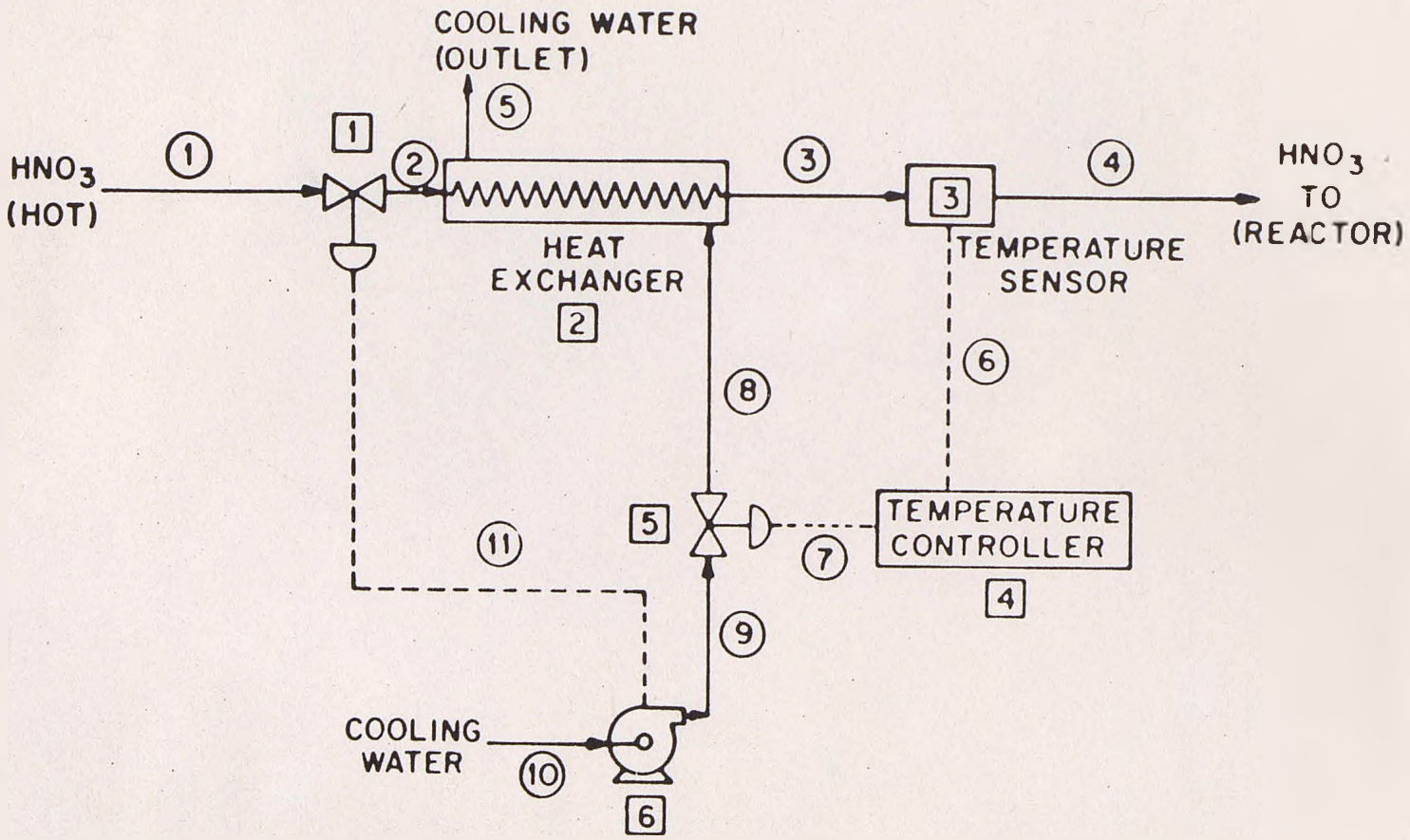
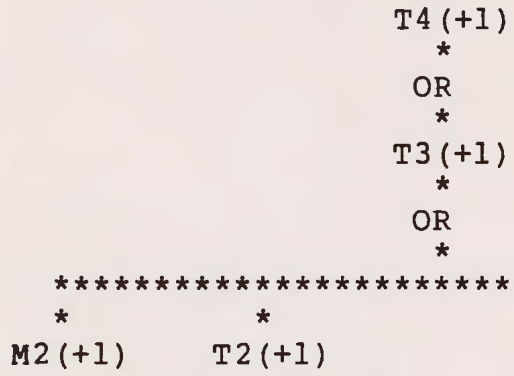


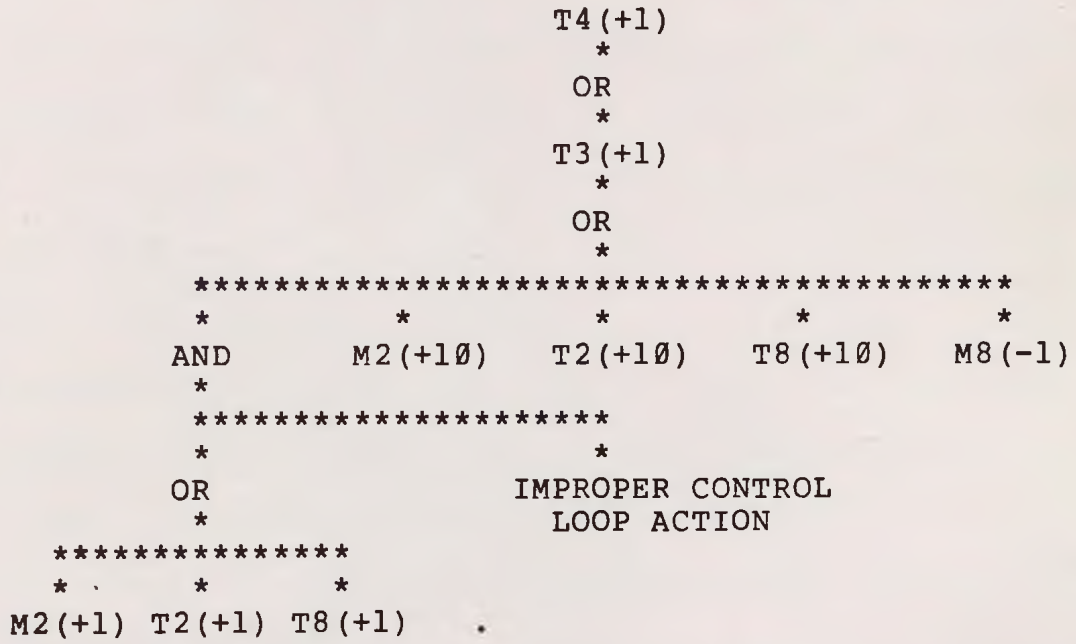
Figure 1: HNO3 Process

His next question would then be "What could cause T3(+1)?" The answer in this case is more complex. From a knowledge of heat exchanger performance, four causes emerge: M2(+1), T2(+1), M8(-1), and T8(+1). A question now arises as to how these causes should be logically connected. Our engineer might reason that since any one of these events would result in T3(+1), the proper operator is an OR gate so his resulting structure would be



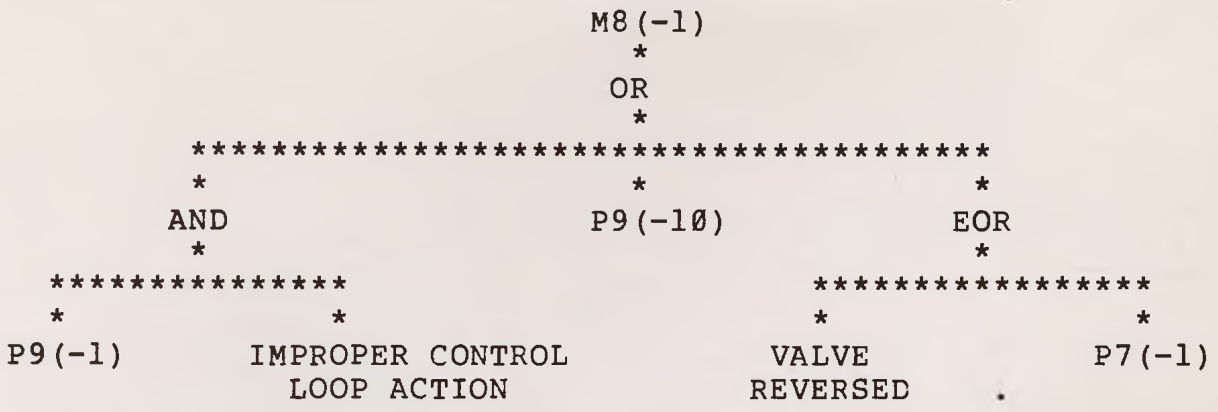
He then checks this tree to see if it is correct. The tree suggests that M2(+1) alone would cause T4(+1). Obviously this isn't true since the negative feedback control loop would act to cancel the effect of M2(+1). Similar arguments can be made to show that T2(+1) and T8(+1) alone will not cause T4(+1). On the other hand, M8(-1) alone will cause the top event. The reason for this is that M8 is itself part of the temperature control loop. Under conditions when T4(+1), the appropriate response for the control loop is M8(+1). M8(-1) is an indication that the control loop is actually working to promote the top event instead of cancelling it out, and this is why M8(-1) alone will cause T4(+1). This

discussion suggests that if either M2(+1), T2(+1), or T8(+1) is to cause T4(+1), the control loop must either take no action to cancel the disturbance or promote it. If deviations in M2, T2, or T8 are very large, however, the control loop may not be able to cancel them out. Hence, these large deviations alone would cause T4(+1). With these arguments, the following tree results



He continues by focusing attention on M8(-1). Assuming that the cooling water control valve is AIR TO OPEN, possible causes are P7(-1) or P9(-1). Suppose, however, that this assumption is incorrect ie. the valve has been installed in the reverse acting mode. In this case P7(+1) will be the cause but note that P7(+1) is a normal response to T4(+1). Hence, three causes really exist: P7(-1), P9(-1), and VALVE REVERSED. Considering that P7(-1) and VALVE REVERSED are on the control loop and applying the logic suggested when

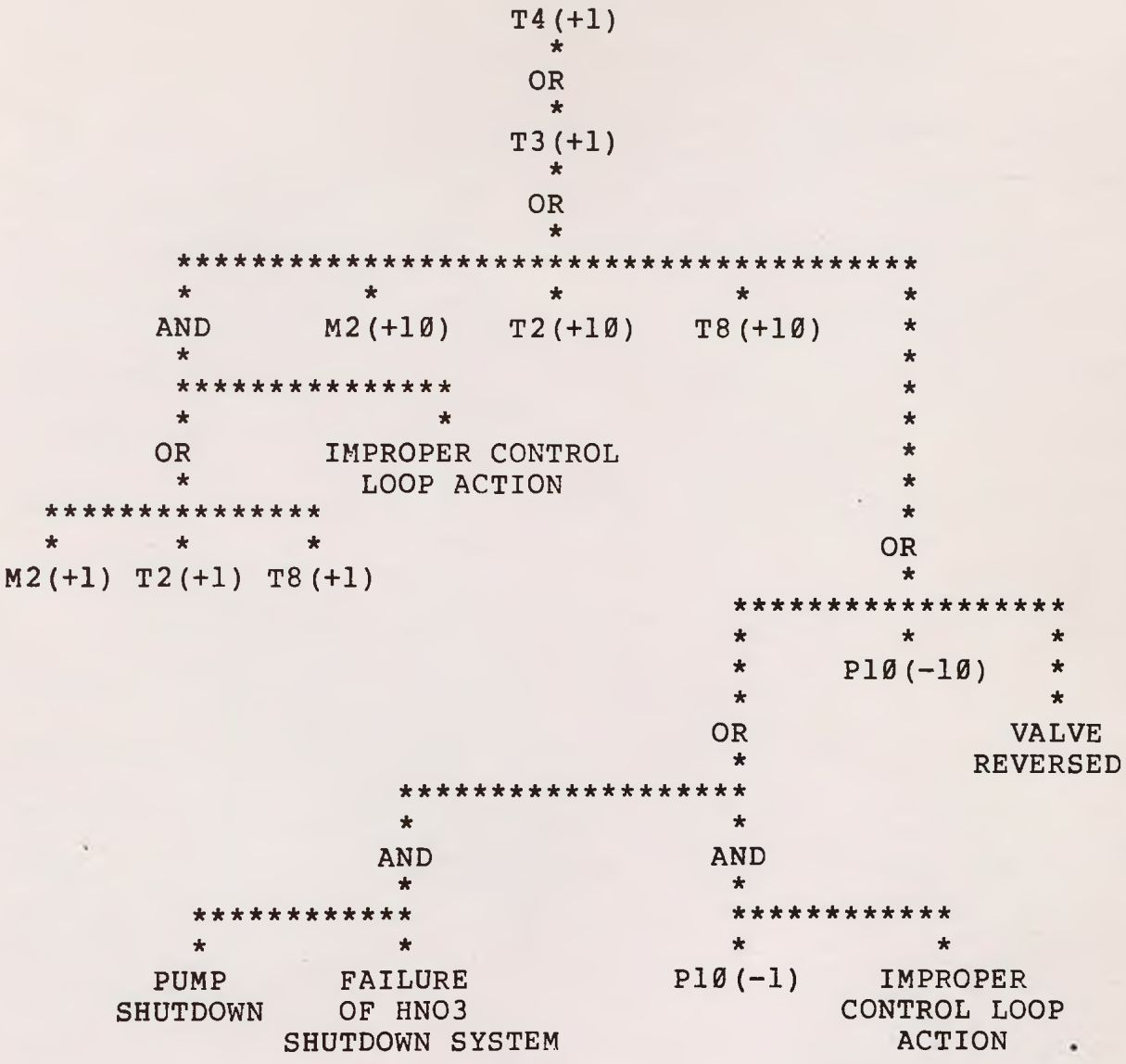
expanding T3(+1), the following expansion for M8(-1) results



The EOR gate is necessary because P7(-1) and VALVE REVERSED together cancel one another out resulting in M8(+1).

Proceeding with P7(-1), our engineer might trace this to P6(-1) and subsequently to T4(-1). This, however, cannot be since he knows T4(+1) to be the case. T4(-1) is a consistency violation and must be dropped from consideration. These are common when negative feedback loops (such as control loops) are encountered in a process. One must be careful to exclude any events which are consistency violations in further analysis. If T4(-1) is the only cause of P6(-1) then P6(-1) must also be dropped, and so on with P7(-1). Continuing the analysis, he might trace disturbances in M2, T2, and T8 to similar disturbances in M1, T1, and T10. Decreases (-1 or -10) in P9 can be traced to either decreases in P10 or a shutdown of the pump. Although PUMP SHUTDOWN stops the flow of cooling water, it also activates a system which stops the flow of hot HNO3. Hence, in order for PUMP SHUTDOWN to cause T4(+1), a failure

of the HNO3 shutdown system is also required. With this, the final tree becomes



The analysis presented here is highly simplified in that many events have been excluded from the tree. It does, however, demonstrate the basic method of manual fault tree construction. Later in this thesis, a more detailed tree for this same system will be constructed using an explicit fault tree synthesis algorithm.

MODELLING OF PROCESS COMPONENTS

In the previous section we saw how one might go about constructing a fault tree for a particular process. At each step, the engineer had to make two basic decisions before building each subtree, the first being a determination of the local causes of each event and the second a decision concerning how these causes should be logically connected.

In order to determine the local causes of an event, one must have some type of model which relates cause and effect. This information can be captured by developing a cause and effect model for each piece of equipment encountered in the process. Let us try to develop some means of modelling cause and effect.

Consider the heat exchanger used in the previous section. We might begin by taking some output or effect variable and listing all variables which could be potential causes of any deviations in that variable. Let us take T3. Some variables which affect T3 are T2, M2, T8, M8, area, and overall heat transfer coefficient. In some cases, the surroundings variables may also have a significant effect on T3. Such variables include the surroundings temperature and velocity, external area, and external heat transfer coefficient. We need some other information to determine how strongly these variables affect T3. The additional information required is the gain between T3 and its causitive variables. The gain between T3 and some variable X is given by $\partial T3/\partial X$ and is a measure of how strongly X affects

T3. In many cases, this gain is quite small and hence, one might say that no relationship exists.

The gain may be computed from performance equations for the specific piece of equipment or estimated from data or one's physical knowledge. For instance, without explicit computation, one could reason that in many cases the gain between T3 and the surroundings velocity over the exchanger is nearly zero. Let us now compute the gain for T3 with respect to T2. A performance equation which could be employed to yield this information is the energy balance for the HNO3 side

$$(A) \qquad M_{23}\, C_{p23}\, (T3 - T2) + U\, A\, \Delta T_{lm} = 0$$

One could either explicitly or implicitly differentiate the equation to obtain $\partial T3/\partial T2$. A problem now arises since the differentiation carries with it the assumption that all other variables remain constant. If T2 is perturbed then T5, the other outlet temperature will also change hence we need to take $\partial T3/\partial T2$ with T5 varying. A convenient way of taking partial derivatives involving more than one dependent variable involves the use of Jacobians [10]. In order to use them, one must have N independent equations for N dependent variables. For the heat exchanger with the specified input/output combination, we will need two

independent equations. This is because if T2 is perturbed, we expect deviations in both T3 and T5. One equation has already been defined and the other we need is simply the energy balance for the water side or

$$(B) \quad M_{85}\, C_{p85}\, (T5 - T8) - U\, A\, \Delta T_{lm} = 0$$

Now $\partial T3/\partial T2$ with T5 also varying is defined as the Jacobian

$$-\frac{\frac{\partial(A,B)}{\partial(T_2,T_5)}}{\frac{\partial(A,B)}{\partial(T_3,T_5)}} = -\frac{\begin{vmatrix} A_{T2} & A_{T5} \\ B_{T2} & B_{T5} \end{vmatrix}}{\begin{vmatrix} A_{T3} & A_{T5} \\ B_{T3} & B_{T5} \end{vmatrix}}$$

where $A_x = \frac{\partial A}{\partial x}$ and the vertical bars indicate the determinant of the matrix. Computing $\partial T3/\partial T2$ in this manner is equivalent to differentiating both equations with respect to T2 and solving for the unknowns $\partial T3/\partial T2$ and $\partial T5/\partial T2$. By using Jacobians, we can compute real gains from performance equations which describe the given piece of equipment.

In addition to gains, dynamic parameters for the various cause and effect relations can also be obtained. Just as gains are necessary to provide information on the relative magnitude of deviations in the cause and effect

variables, dynamic parameters are necessary to obtain data on the time needed to transmit these deviations. For purposes of fault tree analysis, a dead time and first order time constant are often sufficient to describe the dynamics. Both of these parameters may be obtained from unsteady state performance equations. For the heat exchanger, the dead time for the T3-T2 relation is given by

τ_D = (tube length) / (velocity through tubes).

If we approximate the heat exchanger as a mixed tank, the first order lag is given by

$$\tau = M\ CP\ /\ U\ A$$

M = Mass of fluid in exchanger tubes

CP = Specific heat of tube side fluid

U = Overall heat transfer coefficient between shell and tubes

A = Area for heat transfer between shell and tubes

Let us now discuss the form of these cause and effect models. A convenient way to capture cause and effect relations is to use directed graphs (digraphs). Digraph models of equipment are composed of nodes which represent process variables associated with the particular piece of equipment. Cause and effect relations are shown by drawing a directed edge from the cause to the effect variable. Associated with each edge are a gain, dead time, and first order lag as discused previously. A simple digraph model relating cause and effect for the outlet temperature T3 of the heat exchanger is shown on the next page.

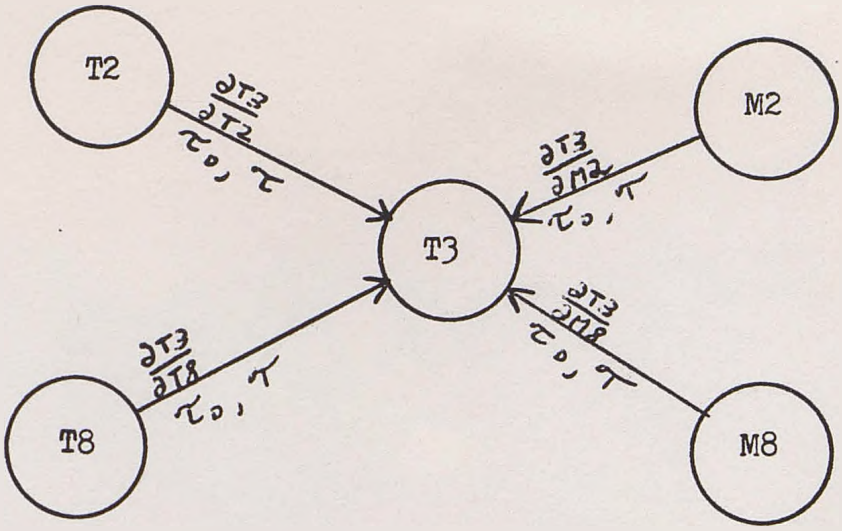


Figure 2: Digraph model for temperature T3

Digraphs can also capture failure modes. A control valve model will illustrate this.

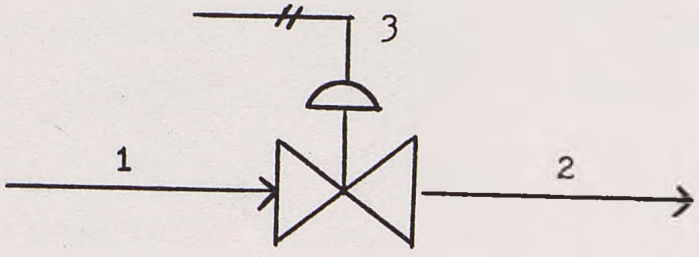


Figure 3: Control valve system

Consider the mass flow out of the valve. We can establish two causes for deviations in this variable these being deviations in the mass flow into the valve and the pressure

on the valve bonnet. The following digraph captures these relations.

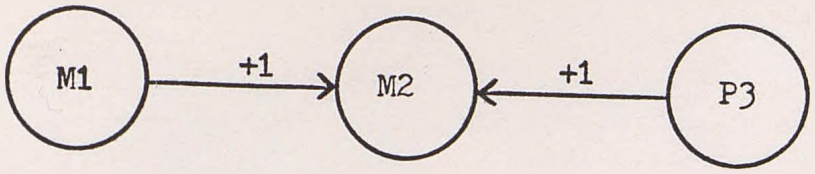


Figure 4: Digraph for mass flow out of control valve

In place of a gain, dead time, and first order lag, the graph has discretized gains of +1 on the edges. These discretized gains indicate that moderate deviations in the cause variable produce moderate deviations in the effect variable. The sign indicates the relative directions of the deviations. We shall use these discrete gains in cases where the real gain is either unnecessary or unavailable. Let us now discuss how we might model failures. If the valve is stuck in position, changes in P3 have no effect on M2. Similarly if the valve is reversed, changes in P3 will have an opposite effect on M2. These failures are events which change the gain between P3 and M2 by their occurrence. We can capture these events on the digraph by drawing multiple edges with different gains between P3 and M2 and

then labelling those edges which are conditional on certain events. This is shown in the following graph.

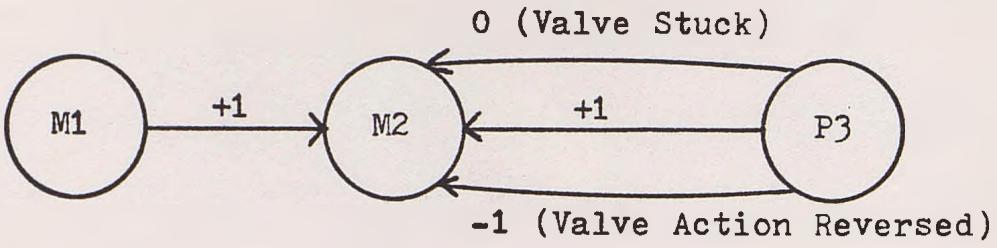


Figure 5: Digraph for mass flow out of control valve

Edges which have no explicit condition on them carry the condition "NORMAL OPERATION". Failures which change relations between variables are modelled using conditional edges. Should a failure cause a deviation in some other variable, this is modelled by representing the failure as a node and then drawing a directed edge from it to the effect variable. Consider the failures "VALVE FAILS OPEN" and "VALVE FAILS CLOSED". These failures cause deviations in M2 and are shown on the following digraph.

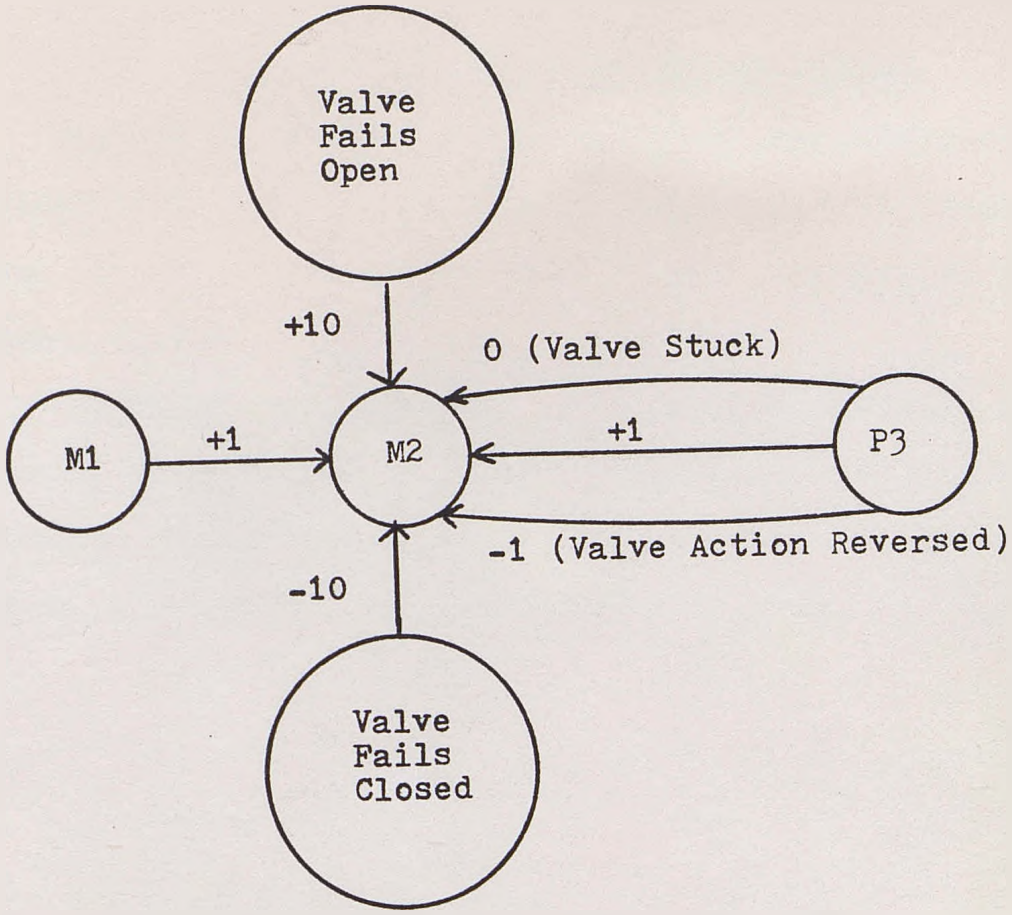


Figure 6: Digraph for mass flow out of control valve

The discretized gains of +10 and -10 indicate that when either of these events occurs, the deviation in the effect variable is large.

Digraph models can be built for valves, pumps, heat exchangers, and even the human operator. They contain both the normal and failed behavior of the components. In the next section, we shall discuss how to merge digraphs for separate pieces of equipment into a system digraph which models the process under study.

DIGRAPH MODELS FOR THE ENTIRE SYSTEM

The previous section dealt with the construction of digraph models for individual components. This chapter will focus on joining these component models together to form the system digraph.

The major consideration in forming the system graph is process connectivity. This information dictates how the component digraphs are to be linked. For instance, if the system is a control valve followed by a pipe, ie.

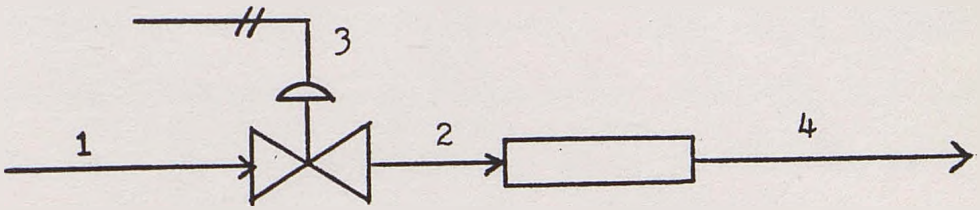


Figure 7: Control valve and pipe system

The digraph for mass flow M4 in the system is given by

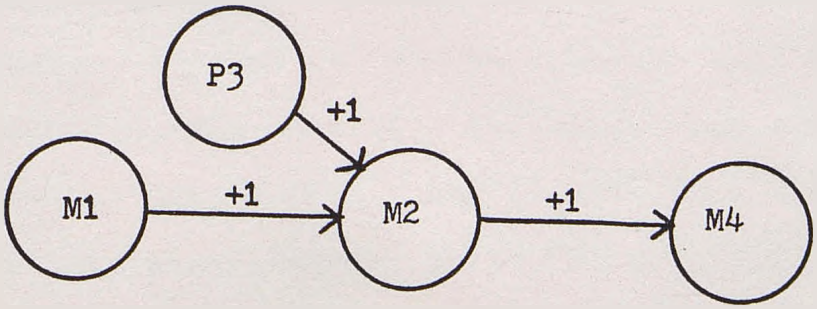
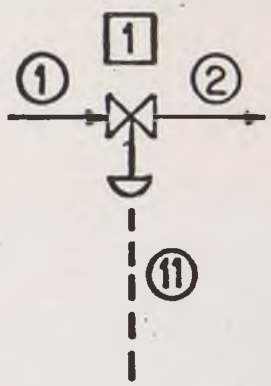


Figure 8: Digraph for M4 in control valve and pipe system

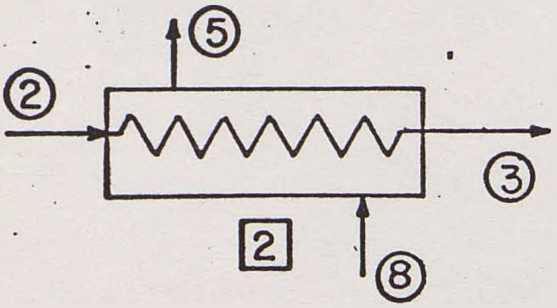
This graph was obtained by linking the control valve and pipe models. The linking process is done by starting at the node representing the top event and then constructing the system graph by working strictly backward from that node. In this way, we can be guaranteed that only those variables which directly or indirectly affect the top event variable are included in the digraph. The basic algorithm for building the system digraph begins with a definition of the top event node. Connections from other nodes are made to this node using those models which have this node as an output variable. Input nodes generated in this way are developed in a similar manner. One rule which must be observed however, is that the model used for the development must not be the same model from which the node was just generated. During the course of digraph construction, it is possible that the same model will be used more than once. Digraph generation proceeds until either battery limits are encountered or until the node to be expanded has already been developed.

As an example, consider the HNO3 flowsheet discussed previously. Figures 9a and 9b contain unit models for the components in this system. The cause and effect relations are shown in matrix form. Let us construct a system digraph for the top event T4(+1) using these models. Note that figures 9a and 9b show only those parts of the models which will be needed for T4 as the top event variable. Let us now construct the graph. Begin with a node defining T4 and ask

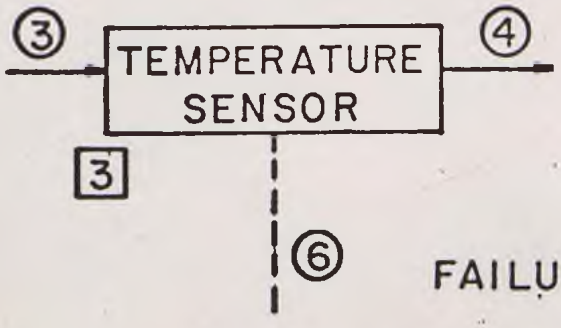


OUT / IN	M 2	T 2
P 1	+ 1	0
P11	-10	0
T 1	0	+1

FAILURE: REVERSED VALVE ACTION	P11 TO M2 GAIN CHANGES TO +10



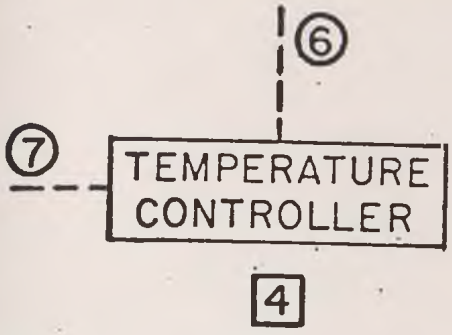
	M5	T3	ETC
M2	0	+1	
T2	0	+1	
M8	+1	-1	
FAILURE: EXTERNAL FIRE	0	+1	



	T4	P6
T3	1	1

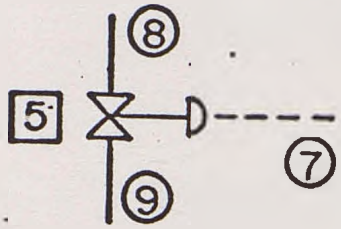
FAILURE: SENSOR BROKEN	CHANGE T3 TO P6 GAIN TO 0 (ZERO)

Figure 9a: Equipment Models For HNO3 Process



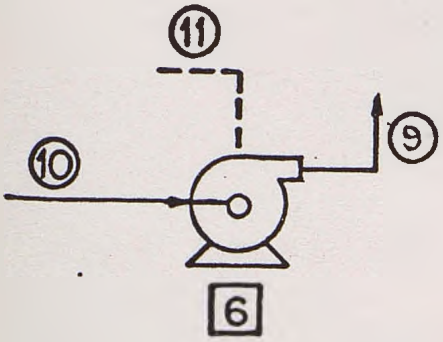
	P 7
P6	+1
FAILURE: EXTERNAL FIRE	+1
LOW AIR PRESSURE	-1

INSTALLED WITH REVERSE ACTION	CHANGE P6 TO P7 GAIN TO -1
CONTROLLER BROKEN	CHANGE P6 TO P7 GAIN TO 0(ZERO)



	M 8
P9	+1
P7	+1

FAILURE: REVERSED VALVE ACTION	CHANGE P7 TO M8 GAIN TO -1



	P9	P11
P10	+1	0
FAILURE: PUMP SHUTDOWN	-10	+1

LINE 11 PLUGGED AT PUMP	CHANGE SHUTDOWN TO P11 GAIN TO 0 (ZERO)

Figure 9b: Equipment Models For HNO3 Process (Cont.)

"Which model has T4 as an output variable?" The answer is the temperature sensor so connect all input nodes to T4 as dictated by this model. One input node is indicated, this being T3 with a gain of +1. The output signal from the sensor (P6) is not considered since it has no direct effect on T4 through the temperature sensor. The graph now has the following form.

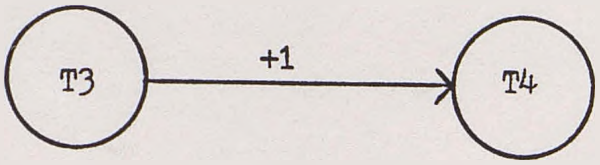


Figure 10: Partial digraph for HNO3 process

Next ask "Which model has T3 as an output variable?" The heat exchanger satisfies this so use it to connect input nodes to T3 as follows.

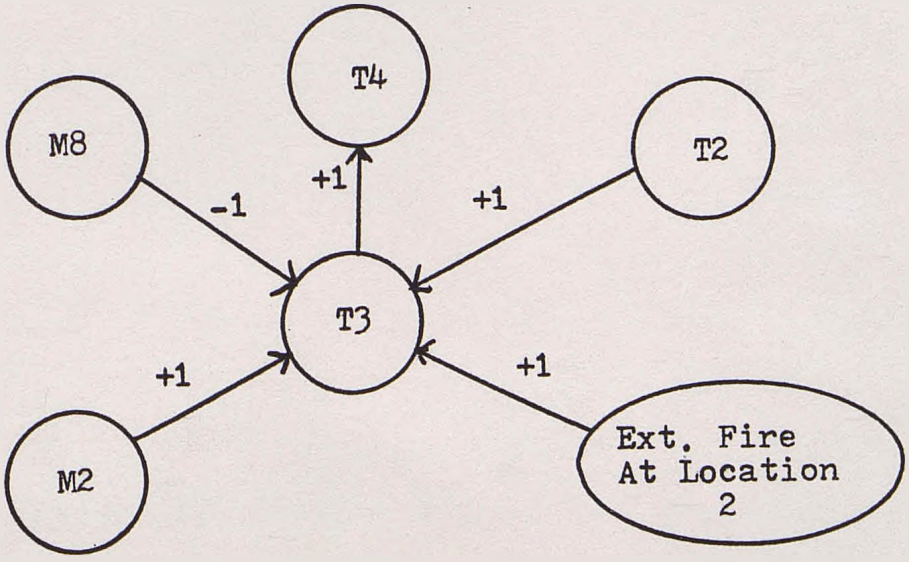


Figure 11: Partial digraph for HNO3 process

Continuing in this manner, develop M8 using the control valve model. Then use the nitric acid shutoff valve to develop both T2 and M2. The graph at this point is shown below.

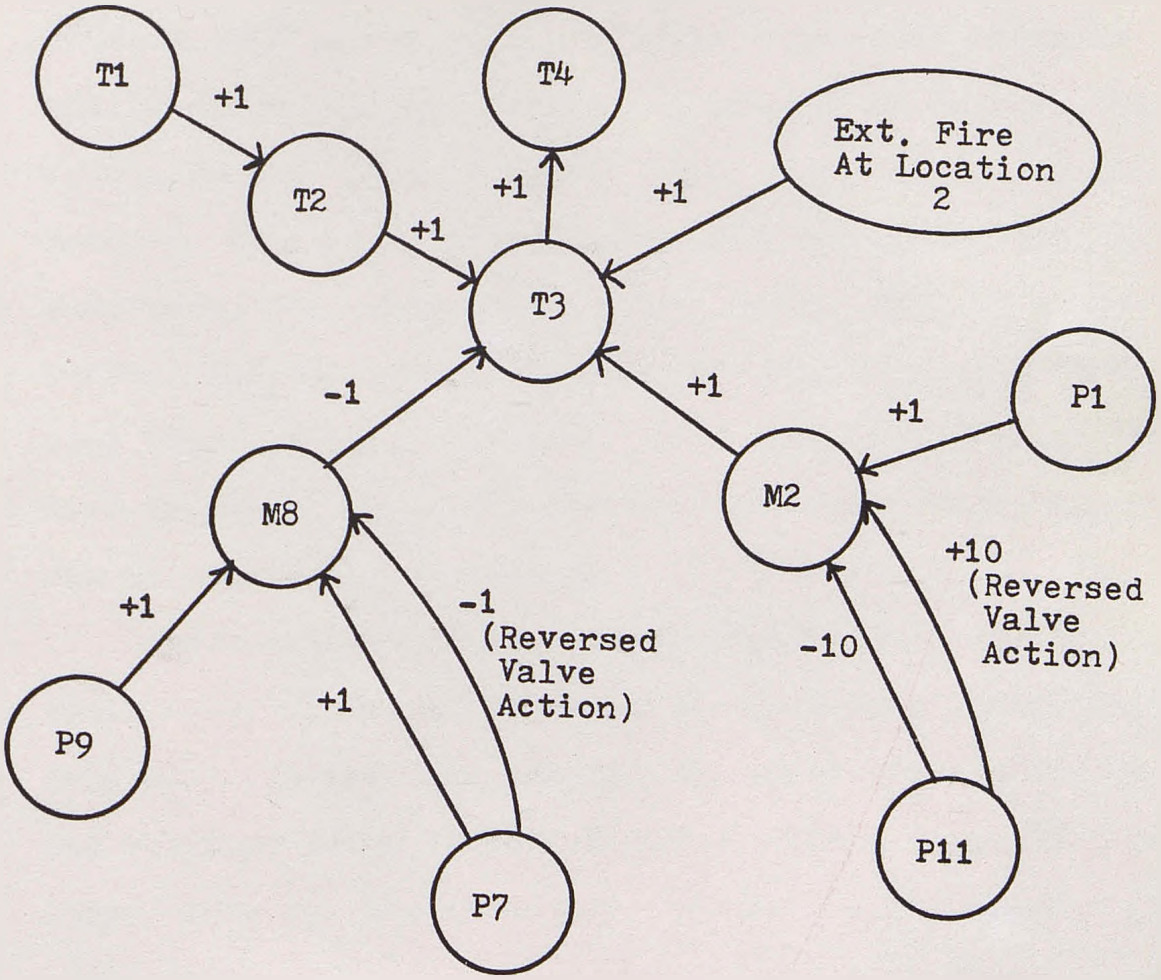


Figure 12: Partial digraph for HNO3 process

Now use the pump model to develop P11 and P9. The controller model is used to develop P7. This generates P6

which is then expanded using the temperature sensor model. The system digraph is now complete and is shown in figure 13.

Some comments on this graph are in order. The expansion for M8 was done using the control valve model. Note that two edges are drawn between M8 and P7. This is because the model used contains the event REVERSED VALVE ACTION which changes the gain between M8 and P7. Similar reasoning is used anywhere more than one edge is drawn between two nodes. The connections between P6 and T3 were made using the temperature sensor model. This model was used twice, however, different output variables were developed each time. After connecting T3 to P6, note that we did not have to trace T3 using the heat exchanger model since this was already done previously.

The next task is to transform the system digraph into a fault tree. This topic will be discussed after the next chapter. In the next section, we shall discuss how to find and classify loops in the digraph. These loops will play a major role in the execution of the fault tree synthesis algorithm.

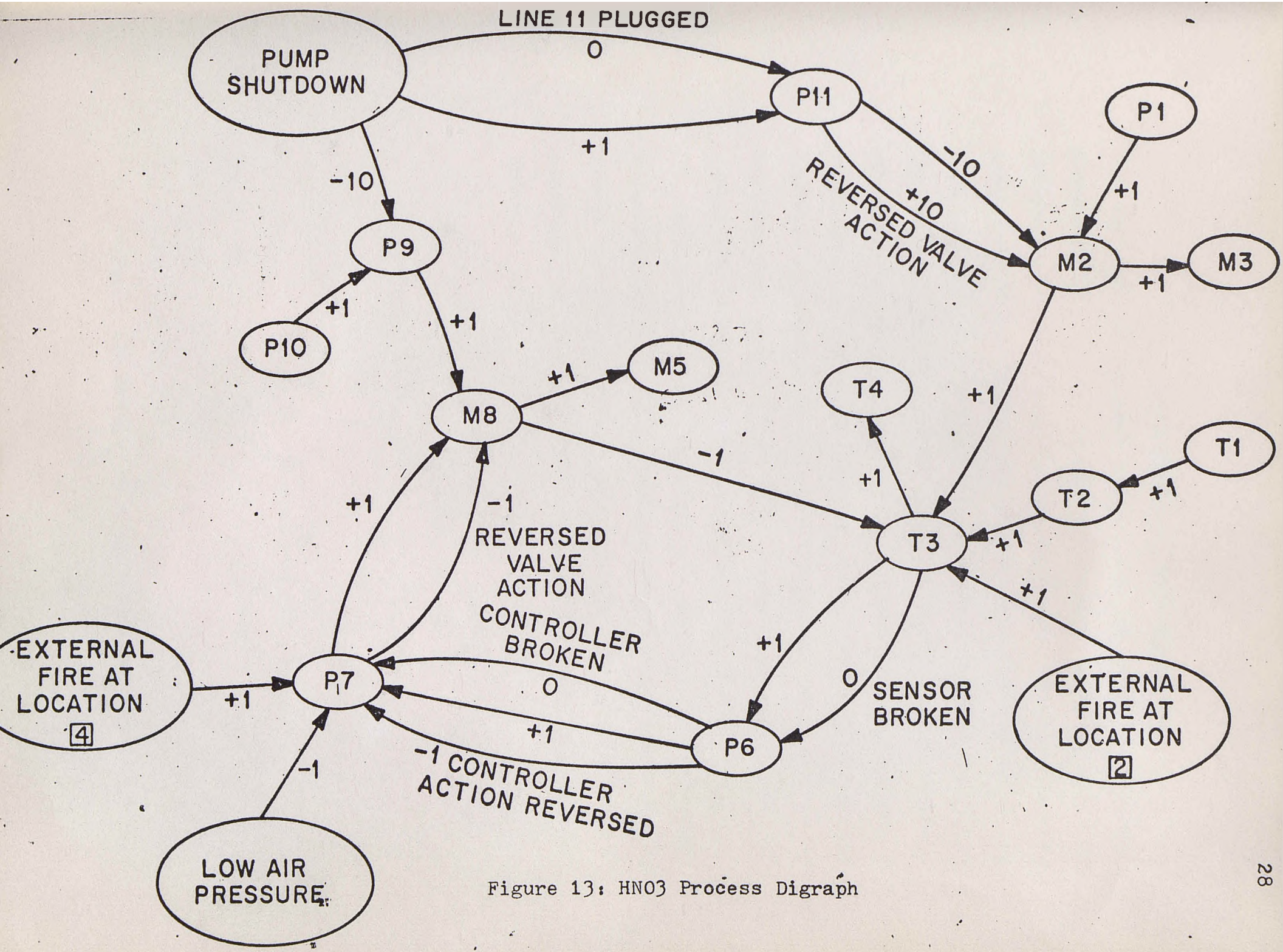


Figure 13: HNO3 Process Digraph

THE IMPORTANCE OF DIGRAPH LOOPS

The system digraph is a map of how disturbances propagate through the system under study. This map contains information on how the process variables interact with one another. Interactions which tend to cancel disturbances are most important in fault tree synthesis. If no cancelling interactions exist, construction of the fault tree consists of simply identifying sources of disturbances and gathering them under an OR gate. Should some cancellation exist, however, we shall have to find conditions under which the disturbances get through the cancelling interaction. All cancelling interactions are loops on the digraph.

The first type of loop is the negative feedback loop. A variable on a negative feedback loop can be traced back to itself on the digraph. The net gain along this path is negative. An example of a negative feedback loop is shown below.

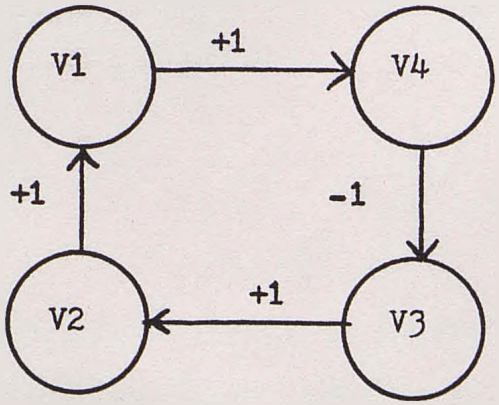


Figure 14: Example of a negative feedback loop

Note that if variable V1 deviates positively or negatively, variable V2 will deviate with an opposite sign to produce a cancelling signal to V1. If the signal from V2 is large enough and arrives back at V1 fast enough to cancel the disturbance, we shall need to find conditions under which this cancellation does not occur.

Normal control loops nearly always appear as negative feedback loops on digraphs. Consider the following pressure control loop.

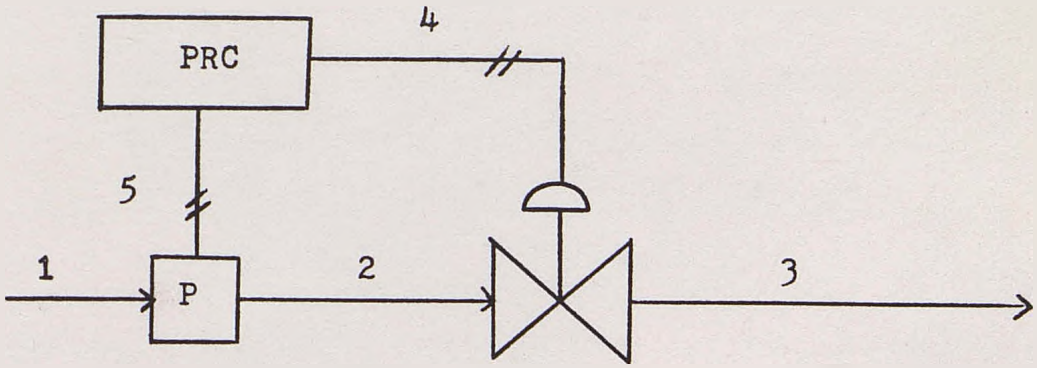


Figure 15: Pressure control loop

Using models similar to those developed in the previous section, we would arrive at the following system digraph with P2 as the top event variable.

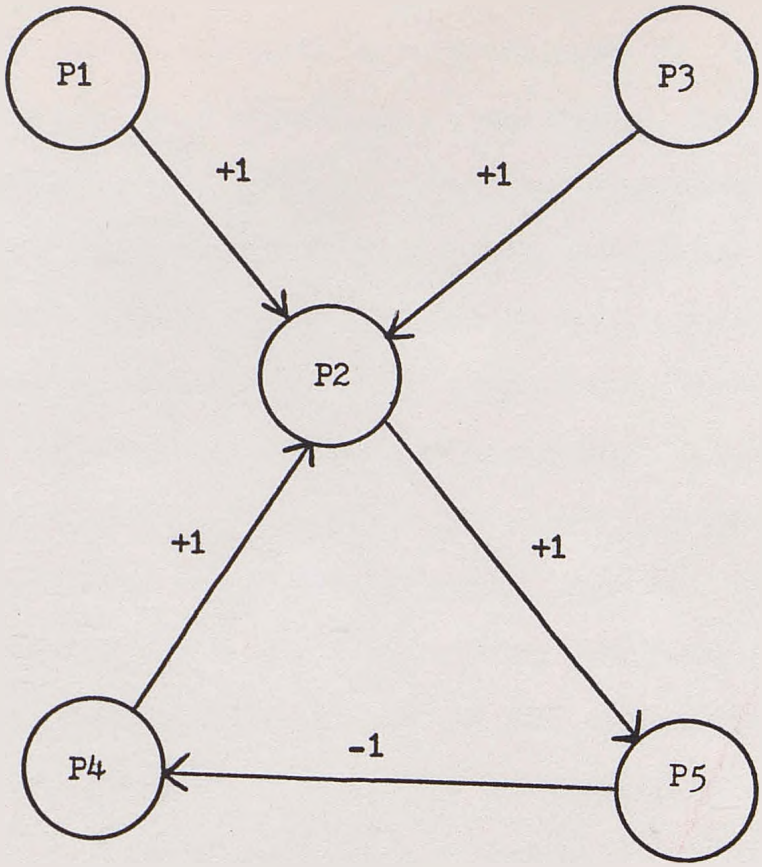


Figure 16: Digraph for P2 in pressure control loop

A negative feedback loop comprised of P2-P4-P5-P2 exists in this graph. If P2 deviates, we expect P5 and subsequently P4 to deviate sending a cancelling signal to P2. Physically this corresponds to the normal action of the pressure control loop.

Negative feedback loops have the characteristic that

they attempt to cancel disturbances which enter them. Whether they actually can or not depends on the strength and speed of the cancellation signal. The strength of the cancellation signal can be determined by computing the net gain around the feedback loop using real gains in the computation. The speed of the loop response may be obtained using the corresponding dead times and first order lags.

When constructing fault trees, use two levels of disturbance, the normal or "1" process disturbance and the alarm condition or "10" disturbance. After a negative feedback loop is found in the graph, determine which level of disturbance, if any, it will cancel. This is done by examining the real gains and time constants around the loop and then determining if there are any elements in the loop which may saturate. Consider the negative feedback loop in figure 16. This loop corresponds to a pressure control loop. If the loop is designed properly, the gains and time constants should be such that normal disturbances are handled properly. Since this is a continuous pressure control loop, we expect it to handle 1 but not 10 disturbances. If the pressure becomes very high at location 2, the valve can only open to its full open position. This saturation indicates that the loop cannot handle a +10 disturbance in P2. Similar arguments apply to a -10 disturbance. Very low pressures cause the valve to fully close and no control action exists after this point.

Not all negative feedback loops are control loops.

Consider a relief valve in a line

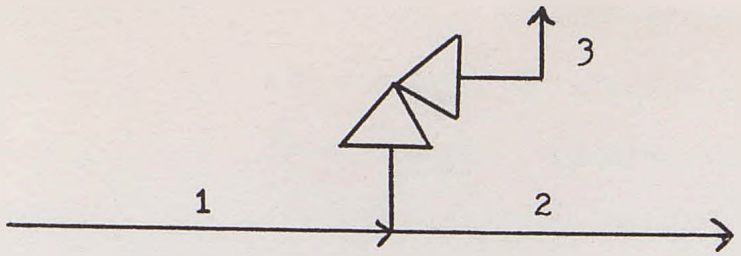


Figure 17: Relief valve in line

The digraph with P2 as the top event variable is shown below.

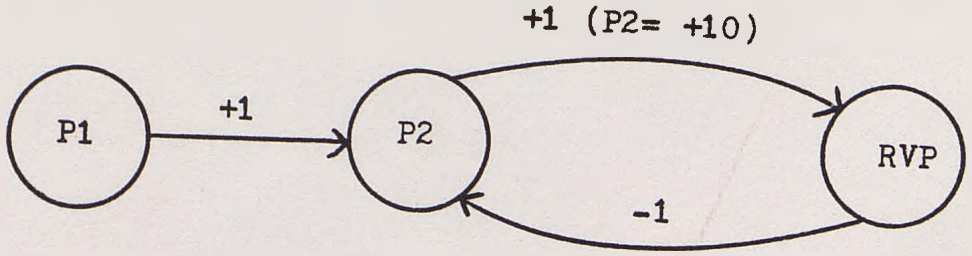


Figure 18: Digraph for P2 in relief valve system

Note that again we have a negative feedback loop through P2. RVP stands for relief valve position. Positive deviations in RVP correspond to the valve opening. If the valve is designed properly, we expect this loop to handle pressure

disturbances (+1 and +10). The edge from P2 to RVP, however, indicates that this loop only activates on +10 deviations in P2. The loop is classified as one which will cancel only +10 disturbances in P2. Physically this simply means that the relief valve should open and relieve the pressure in the line only when P2 becomes very high. Normal disturbances in pressure (+1) are not affected by this loop.

Some negative feedback loops naturally occur in certain processes. Examples of this are endothermic chemical reactions and the negative temperature coefficient in a nuclear reactor. In the first case, higher temperatures cause higher reaction rates which tend to lower the temperature. The digraph for an endothermic reaction is shown on the next page.

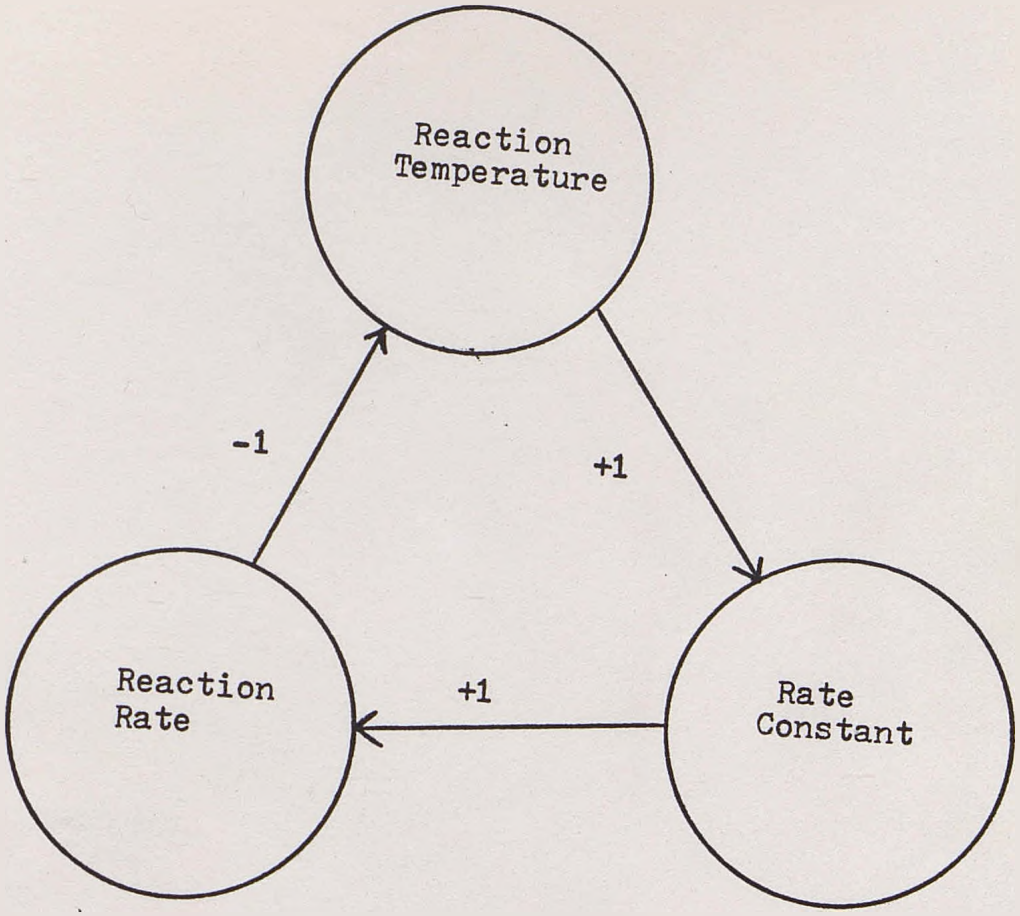


Figure 19: Digraph model for an endothermic reaction

In the second case, higher temperatures result in a lowering of the reaction rate due to a decrease in the cross section. This results in lower temperatures. The digraph which captures this is shown on the next page.

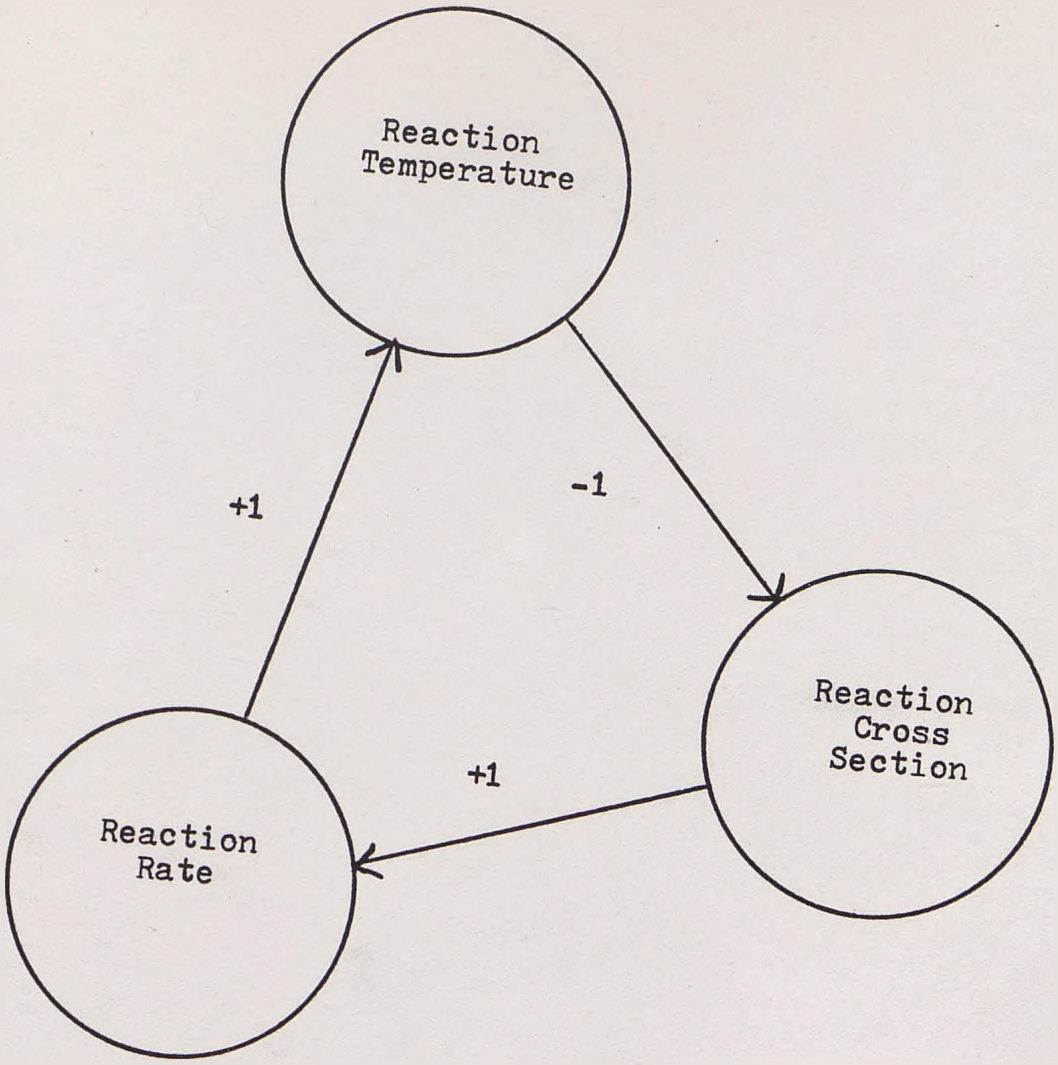


Figure 20: Digraph for a nuclear reaction

Not all feedback loops are negative. In cases where the net gain around the feedback loop is positive, special consideration may have to be given to the loop. If the loop has an active element within it and can drive itself to an extreme, it must be regarded as a disturbance source. This is because noise will usually cause this type of loop to go

unstable. A good example is an exothermic reaction for which the digraph is shown in figure 21.

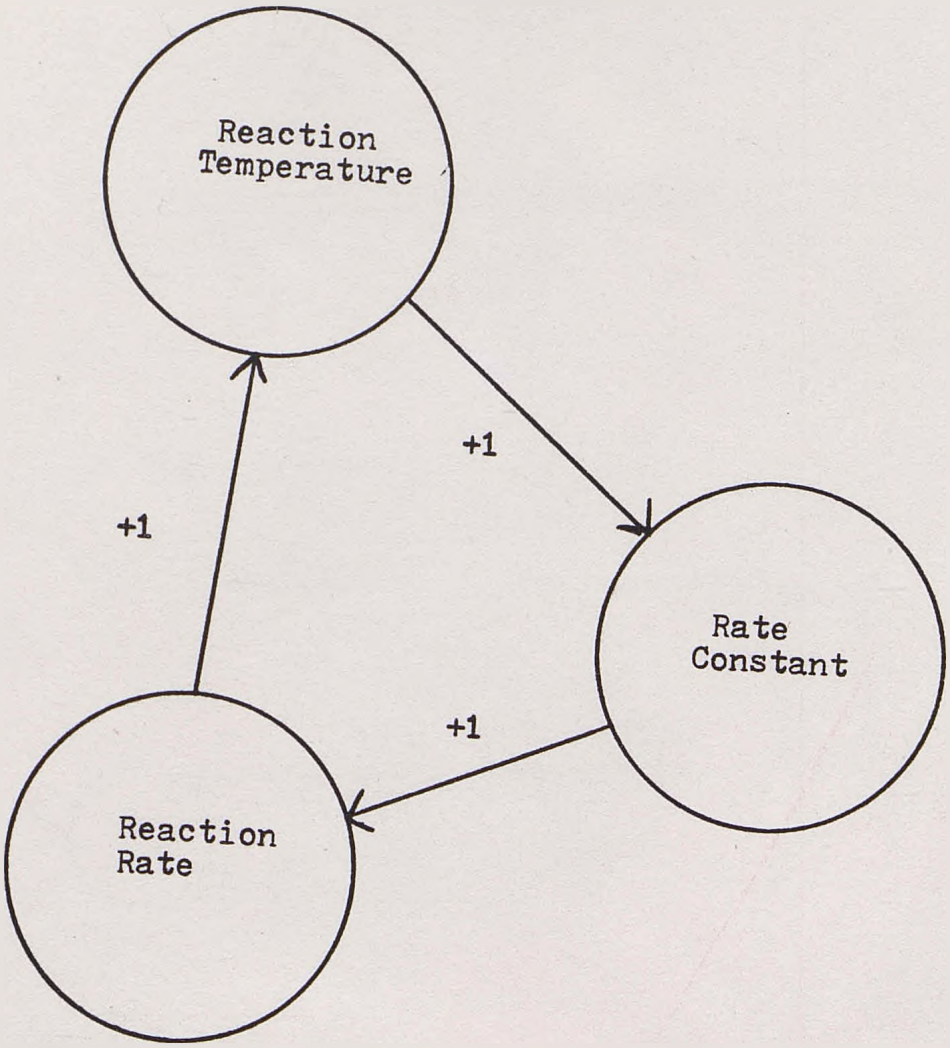


Figure 21: Digraph model for an exothermic reaction

The active element in this case is the reaction itself which draws energy from the bonds within the molecules. If no active element exists within the loop, the loop will not drive itself to an extreme. An example of this type of loop

occurs in the digraph description of the pressures in a series of pipes.

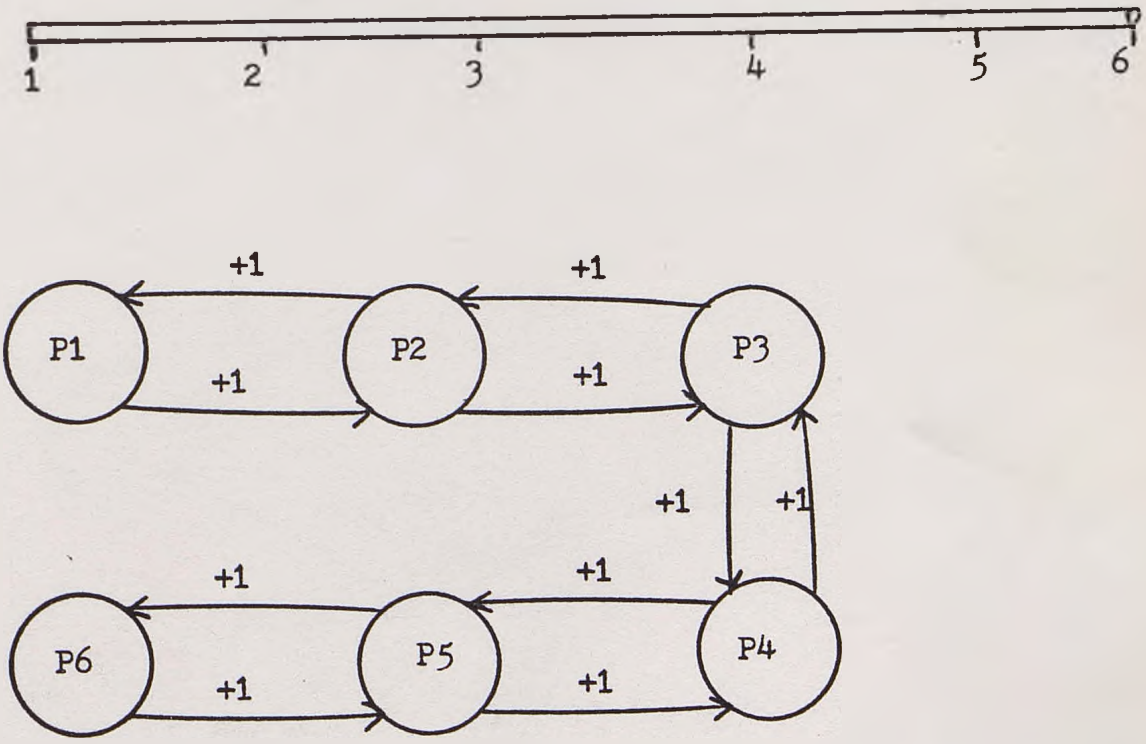


Figure 22: Pipe system with digraph model for pressure

A deviation in any one of these pressures will not drive the entire system to an extreme. Positive feedback loops of

this type indicate that information may flow to and from all variables which are on the loop. Deviations in any one of the variables may be sensed by examining any of the variables on the loop. Mass flows in a pipe may have a similar digraph.

Another important type of interaction is the negative feedforward loop. Negative feedforward loops occur when one variable affects another on different paths with opposite net gains on the alternate paths. The general structure is shown below.

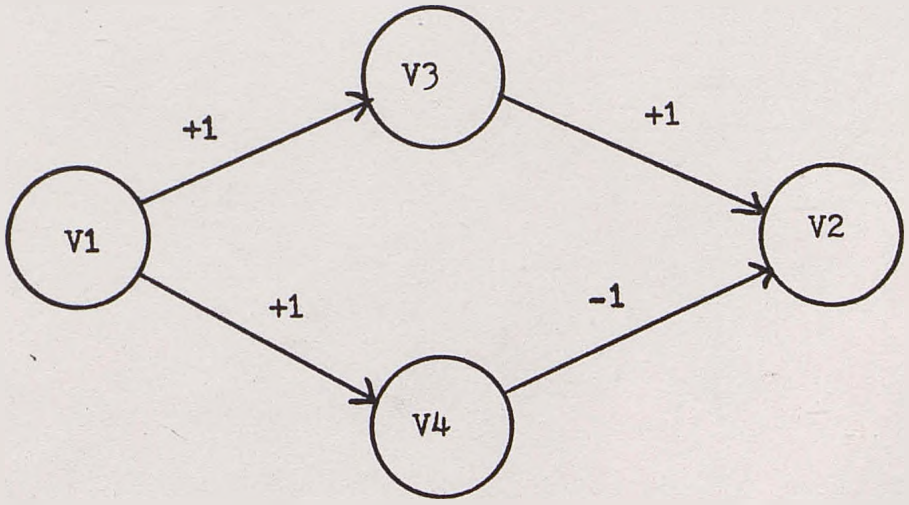


Figure 23: Example of a negative feedforward loop

Should V1 deviate, it will tend to produce opposite effects in V2 on different paths. If the gains and dynamics on both paths are equal, then the disturbances will cancel and no deviation will result in V2.

Negative feedforward loops generally appear as models

of control systems when the control signal is fed ahead in the process. Consider the following system

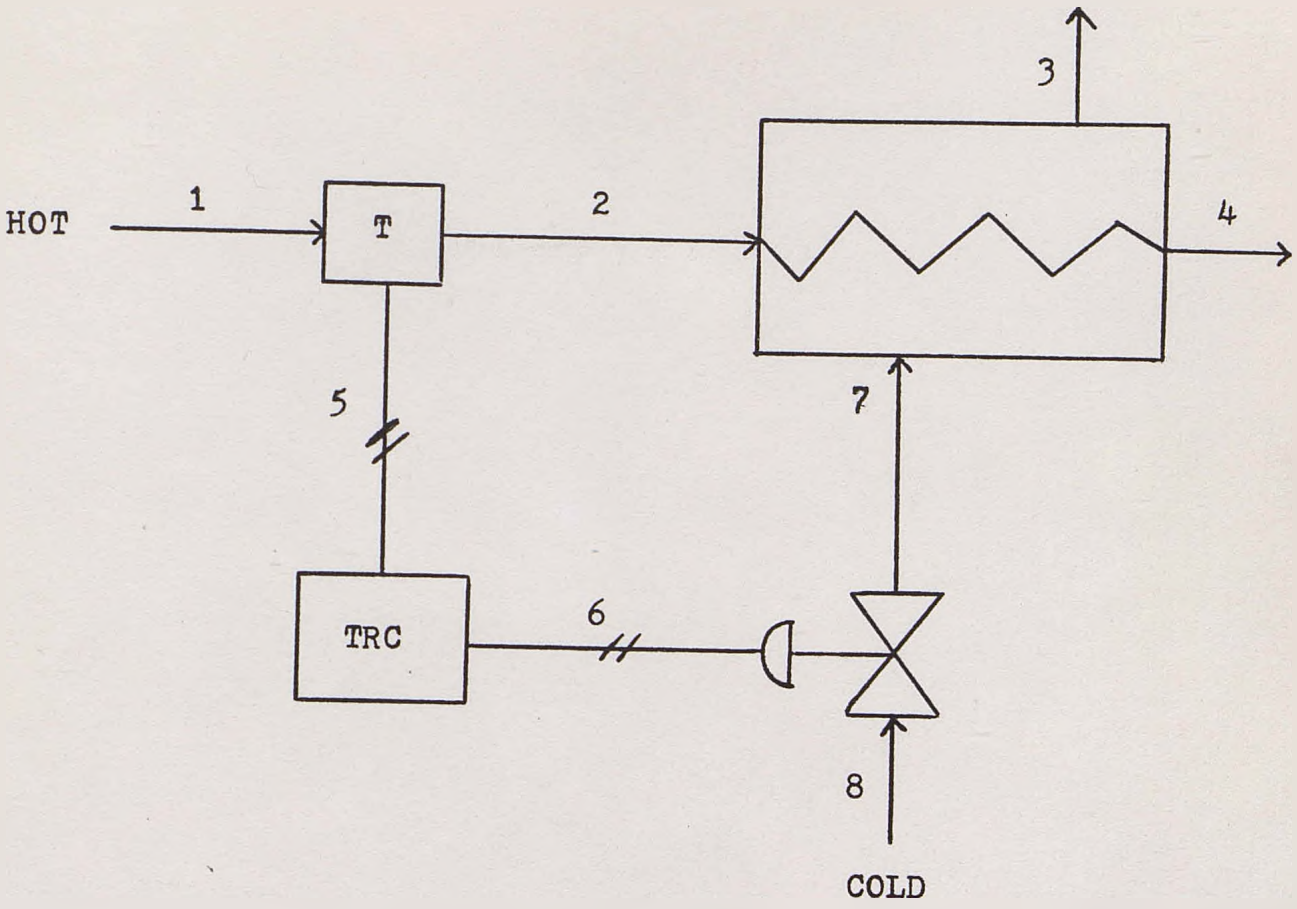


Figure 24: Feedforward control system

If the temperature in stream 1 becomes high, a signal is sent to open the valve and increase the cooling water flow. The system digraph is shown on the next page.

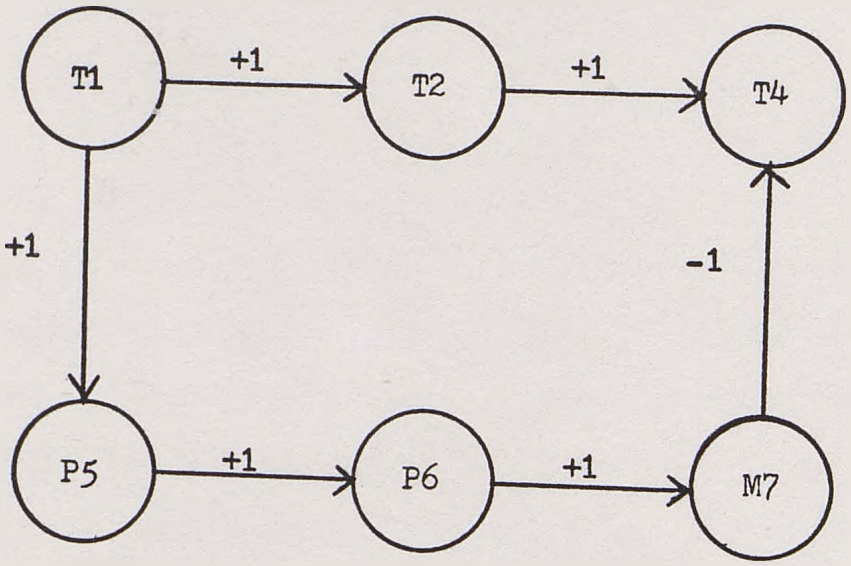


Figure 25: Digraph model of feedforward control system

From the graph, we see that T1 has different effects on T4 depending on which path we trace between these variables. In order to evaluate this loop, we must ask if deviations proceeding down both sides of the loop will cancel one another when they reach T4. Again real gains and dynamics may be used, however, reason that if this control loop is

designed correctly, the disturbances should cancel. If T1 becomes very high (+1Ø), the cooling water valve will saturate to the full open position. Very low temperatures (-1Ø) will cause the valve to fully close. Since no control action exists once the valve is saturated, this loop handles "1" but not "1Ø" disturbances in T1.

Unlike negative feedback loops which cancel deviations no matter where they enter the loop, negative feedforward loops cancel only those disturbances which originate at the common variable which starts the loop. The range of deviation handled by this loop similarly pertains only to deviations in the variable which starts the loop. In order to get a deviation through a negative feedforward loop, it is necessary to transmit the disturbance down one path and then find conditions under which no cancelling signal is transmitted along the alternate path. Positive feedforward loops require no special attention since they cannot drive themselves to an extreme nor do they provide any cancellation effect. Positive feedforward loops do, however, provide more than one path for the transmittance of a disturbance.

Having found and classified loops in the digraph, we are now ready to proceed with the synthesis of the fault tree. The next section describes an algorithm for fault tree synthesis. Following that, we shall use it to construct a fault tree for the HNO3 process.

AN ALGORITHM FOR FAULT TREE SYNTHESIS

The problem of Fault Tree Synthesis may be formulated using a State Space representation. Problems represented in the State Space are solved by transforming a given initial state into a desired goal state. Transformation is accomplished through the use of appropriate operators which transform one state into another. The initial state in Fault Tree Synthesis is a definition of the Top Event along with a description of the process. The process description is in the form of a digraph. The goal state is a Fault Tree connecting the Top Event to events which are not developed further. These events are called Primal Events. The task now becomes one of defining operators necessary to perform the synthesis. In order to do this, consider the methods used in manual Fault Tree construction (See section entitled "Fault Tree Synthesis - A Protocol"). In developing each event, one asks "What could cause this?" On a digraph, this corresponds to asking "Which nodes are inputs to the node representing the current event?" Once these events are identified, the next task is to determine how they should be logically interconnected to form part of a Fault Tree. In answering the question, "What could cause this?", it seems natural to connect the events by using an OR gate. As noted previously, however, this is sometimes incorrect. The problem lies in the fact that the use of an OR gate represents only a partial answer to what the causitive events are. In stating that one event causes another, it is

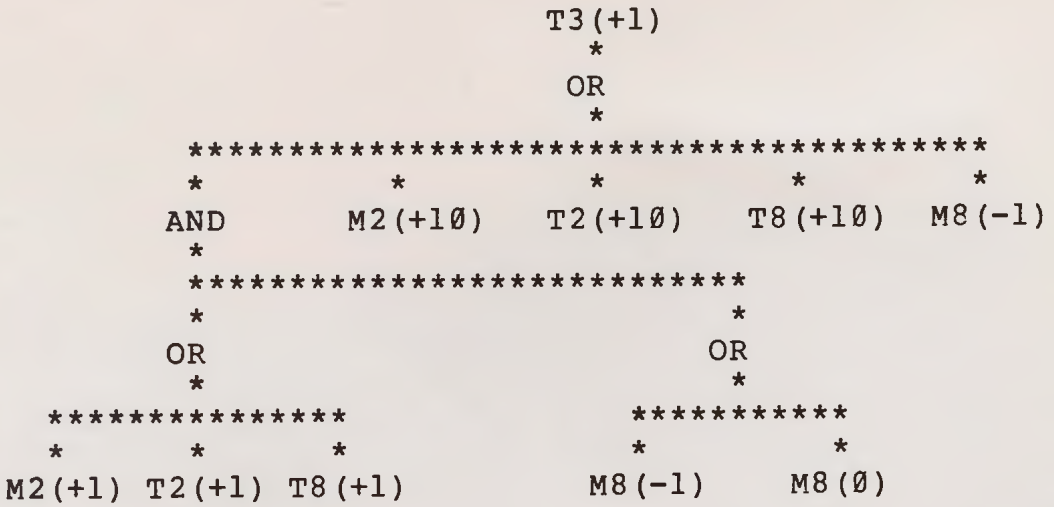
necessary to include the qualification that nothing else happens which will cancel the original effect. This important assumption, although not explicitly stated, is made at each step in the synthesis of the tree.

Constructing a tree requires explicitly stating this assumption at each step. Consider the example used previously. In particular, examine the causes of T3(+1). Recall that this event can be traced to four causes: M2(+1),T2(+1),M8(-1),and T8(+1). If any one of these events is to cause T3(+1), then none of the other variables may deviate in such a manner as to cancel the effect. In expanding T3(+1), one may use the OR gate except that now the inputs to the gate are M2(+1) AND (NOT(T2(-1) OR M8(+1) OR T8(-1))) instead of M2(+1) alone, T2(+1) AND (NOT(M2(-1) OR M8(+1) OR T8(-1))) instead of T2(+1) alone etc. Hence, replace each of the NOTS by its Boolean equivalent structure ie. NOT T2(-1) is the same as T2(+1) OR T2(Ø) etc. This considerably complicates the tree and a simplification is necessary. The event T2(Ø) may be traced to T1(Ø) which means T1 UNCHANGED. T1, however, is a primal variable (deviations in it are primal events). Since primal events are normally considered to have have low probabilities, the event T1(Ø) is nearly always true. A similar argument may be made to show that M2(Ø) and T8(Ø) are also true with high probabilities. Hence the events NOT T2(-1), NOT M2(-1), and NOT T8(-1), are also true with high probabilities and are neglected in the AND gate.

Now consider M8(Ø). Unlike the other events discussed, M8(Ø) is not normally true. Recall that since T3(+1) is true (This is the event being developed.), M8 should increase due to the action of the control loop. Thus NOT M8(+1) is normally not true and remains on the AND gate explicitly (or in its equivalent form of M8(-1) OR M8(Ø)). Note that the event M8(-1) is not even part of an AND gate since all of its COMPLEMENTARY NOTS are normally true. M8(-1) is an indication that the control loop actually is responsible for the disturbance since M8 is the variable which should increase to cancel T3(+1).

This assumption of "ALL OTHER THINGS BEING THE SAME" is a crucial one. In the following developments it is assumed true for all events that are not interconnected by negative feedback or feedforward loops in the digraph model. Hence, it is quite important that the model capture any interactions which might occur in the system.

Consider now, the same events with "1Ø" values. As with "1" values one can argue that T2(Ø),M2(Ø),and T8(Ø) are normally true. Since a "1Ø" indicates a very large disturbance, assume that the control loop cannot cancel it. In terms of these variables, this means that M8 cannot increase sufficiently to overcome the disturbance. Stated otherwise, NOT M8(+1Ø) is normally true. Hence, all of the NOTS associated with the "1Ø" values are normally true so no AND gates are necessary. All of these results are embodied in the following tree for T3(+1).



Note that this tree is the same as that previously developed manually except that M8(-1) OR M8(Ø) has replaced IMPROPER CONTROL LOOP ACTION. This is fine since one of the causes of M8(-1) OR M8(Ø) is IMPROPER CONTROL LOOP ACTION because the loop should act to send M8(+1).

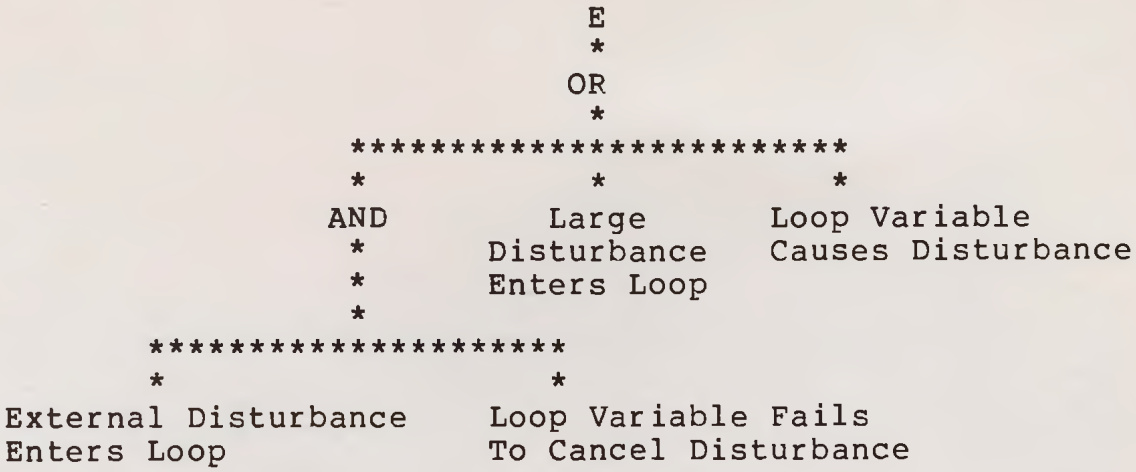
From a functional point of view, note that a disturbance propagates through the control loop if:

(1) The disturbance is extremely large in magnitude.

(2) The disturbance is caused by the control loop itself.

(3) An external disturbance enters the system and the control loop does not act to cancel it.

These statements are general and apply to any negative feedback loop. Hence, they may be used to determine the causes of a disturbance when a negative feedback loop is involved. The generalized operator used for negative feedback loop variables is

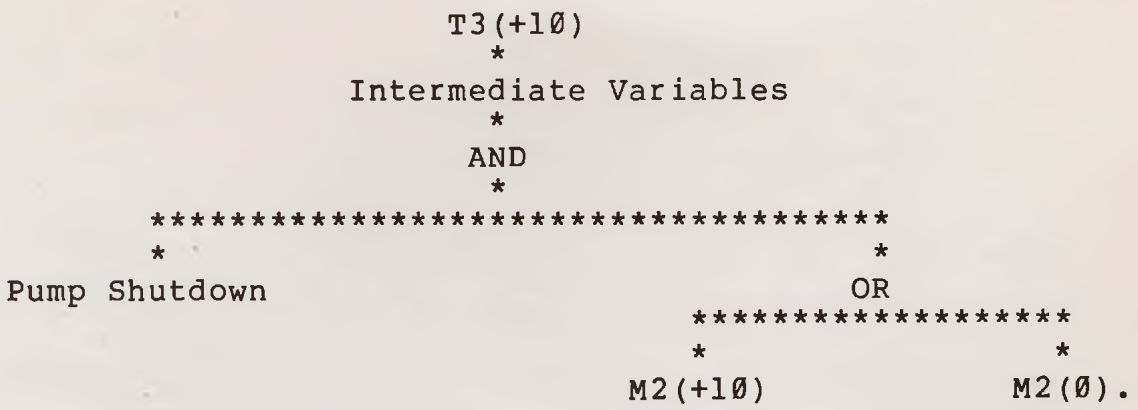


(E represents the event currently being developed.)

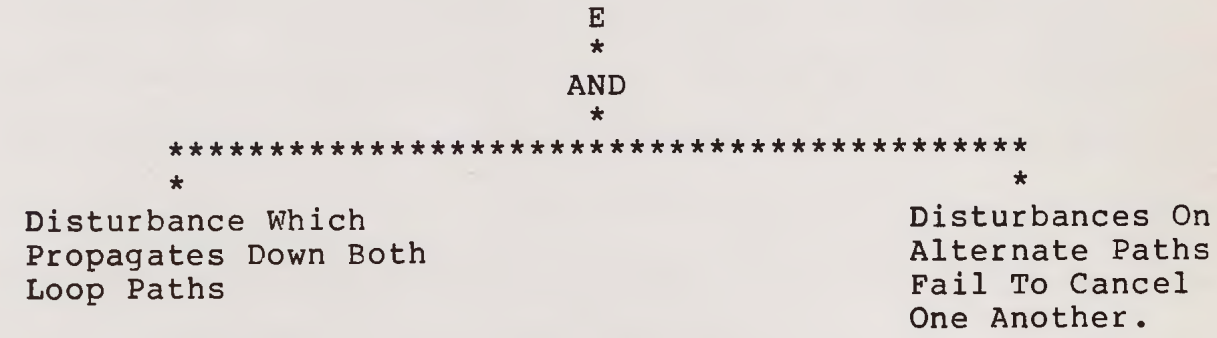
This operator may then be used whenever a negative feedback loop is encountered. Note that the structure of this operator arises from a consideration of how the NOT condition should be handled on a negative feedback loop.

Now consider an operator for negative feedforward loops. This is another case where the "NOT OTHER THINGS CHANGING" assumption is important. A negative feedforward loop is a structure within a process whereby one process variable affects another with opposite gains on different paths within the process. An example of this in the HNO3 problem is the effect of PUMP SHUTDOWN on T3. On one path, PUMP SHUTDOWN tends to send T3(+1Ø) while on another T3(-1Ø). In the former case the sequence PUMP SHUTDOWN-P9(-1Ø)-M8(-1Ø)-T3(+1Ø) occurs while in the latter case PUMP SHUTDOWN-P11(+1)-M2(-1Ø)-T3(-1Ø). Thus to trace T3(+1Ø) to a PUMP SHUTDOWN, one must include the additional condition that disturbances on the other paths of the loop are not present. In this particular problem,the condition

is embodied in NOT M2(-1Ø). Hence, the tree in this case would look like



The generalized feedforward loop operator is then



(E represents the event currently being developed.)

Note that this operator is applied when the point common to both sides of the loop is reached.

For events which involve neither negative feedback nor feedforward loops, the operator used is quite simple. It is just an OR gate since, in the absence of the negative loops, all of the NOT conditions are normally true.

Generalizations of these operators have been made to handle variables which are on combinations of negative feedback and feedforward loops. Given these operators, the local cause and effect relations in the process, and a list of the negative feedback and feedforward loops, it is

possible to construct Fault Trees. The digraph is useful here. Local cause and effect relations are embodied in the edges connecting the nodes. Negative feedback and feedforward loops may be determined by inspection of the graph.

Another important feature in Fault Tree Synthesis is consistency. Consistency means that two mutually exclusive events cannot occur at the same time. For example, in the HNO3 problem, consistency would require that T3(+1) not be traced to T3(-1) or T3(Ø). Any inconsistent events that are generated in the course of the synthesis must be deleted. One might conclude then that any generated events must be checked for consistency against all events which have been developed. Fortunately, this is not true. It turns out that consistency is only a problem when negative feedback or feedforward loops are involved. Hence, the only events which need to be checked are those which involve either of these types of loops. In the case of feedback loops, the event being developed is stored for later consistency checks. The event common to all paths of the feedforward loop is stored when the feedforward operator is applied.

Presented below is a general algorithm based on the concepts developed in this section. In the next section, this algorithm is applied to the HNO3 system.

Fault Tree Synthesis Algorithm

1. Generate the digraph, then find and classify all negative feedback and negative feedforward loops.

2. Select node representing Top Event.

3. Determine local causes of this event by noting the inputs to the node of the digraph.

4. Delete any local causes which violate consistency.

5. Select the appropriate operator depending on whether negative feedback or feedforward loops pass through the current node. Use this operator to logically connect the remaining local causes. If negative feedback or feedforward loops are involved, store the appropriate event for later consistency checks.

6. Select a node corresponding to an undeveloped event and return to step 3. If only primal events remain, stop.

ALGORITHMIC FAULT TREE CONSTRUCTION

The problem is to synthesize a fault tree for the event T4(+1) (T4 high) for the HNO3 process shown in figure 1. The system digraph has already been generated and is shown in figure 13. This figure is reproduced on the following page.

Begin with loop identification and classification. A negative feedback loop exists through the variables T3-P6-P7-M8-T3. The variables are elements of the temperature control loop so assume the loop has been designed to handle normal disturbances. If the disturbances are large enough in magnitude, the cooling water control valve will saturate to either the full open or closed position. After this, no control action exists. Classify this loop, therefore, as being able to handle 1 but not 1Ø disturbances. A negative feedforward loop also exists in this graph. This loop has two paths these being PUMP SHUTDOWN-P9-M8-T3 and PUMP SHUTDOWN-P11-M2-T3. Evaluate both paths of this loop. PUMP SHUTDOWN causes a rapid decrease in M8 but T3 will not rise immediately since there is some capacitance in the heat exchanger. On the other path, PUMP SHUTDOWN should send a signal to P11 almost immediately. If the valve is designed properly, it should halt the nitric acid flow quickly. The PUMP SHUTDOWN-P11-M2-T3 path should, therefore, have enough gain and speed to cancel a disturbance on the other path. Hence, classify this loop as being able to handle disturbances

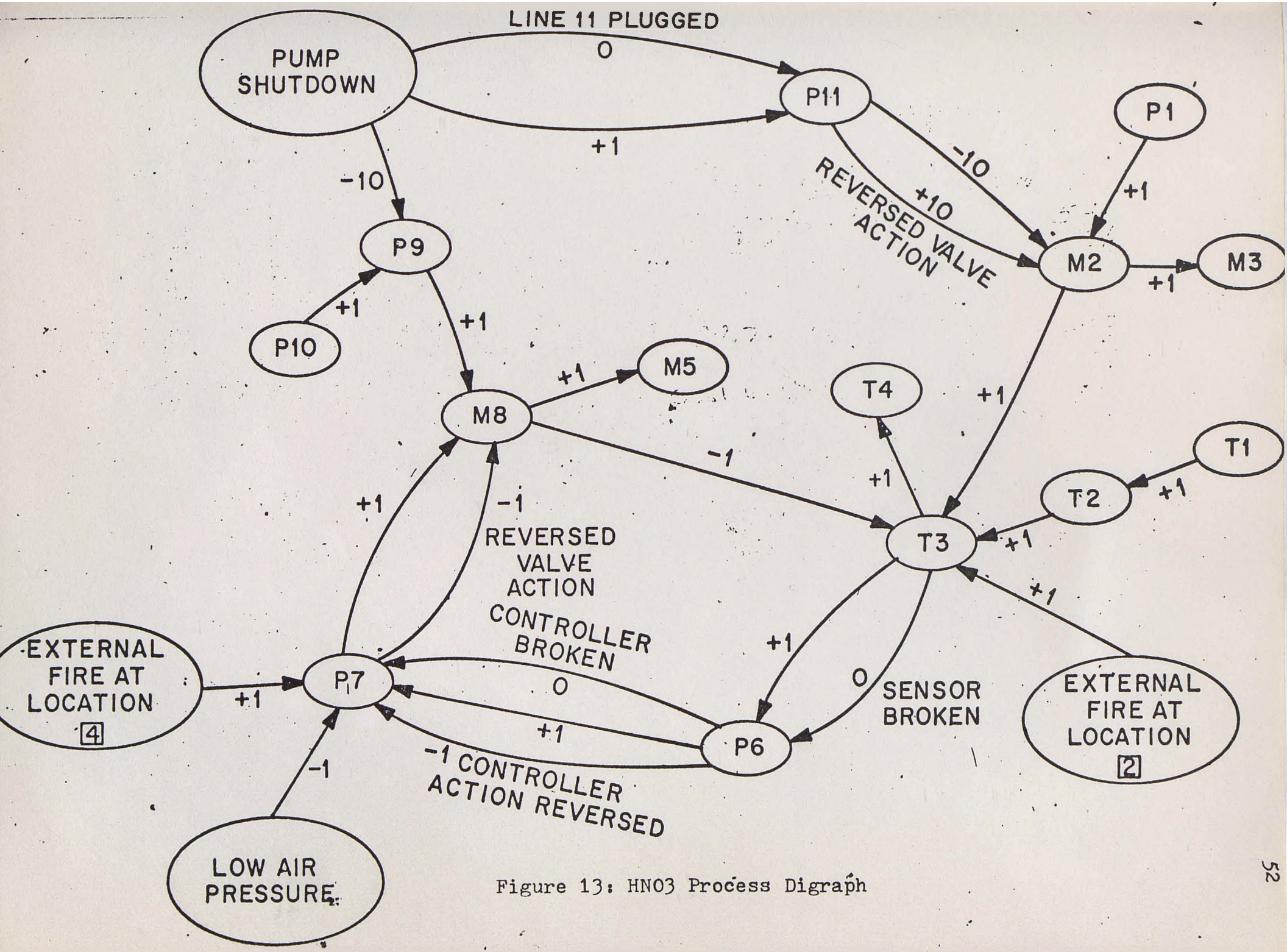


Figure 13: HNO3 Process Digraph

originating from a shutdown of the cooling water pump.

Now begin with the construction of the tree. T4(+1) is the top event. The local cause of T4(+1) is T3(+1). This was determined by noting that T3 is the input to T4. The +1 value for T3 was computed by taking the value of T4 and dividing it by the gain between the nodes. In general,

INPUT VALUE = OUTPUT VALUE / GAIN.

Since T4 is not on any type of loop, use an OR gate as the operator and get the following tree.

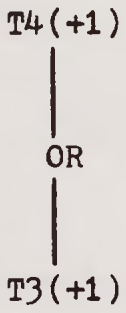


Figure 26: Partial fault tree for the HNO3 process

Next take T3(+1). Local causes from the graph are M8(-1), M2(+1), T2(+1) and EXTERNAL FIRE AT LOCATION 2 (+1). EXTERNAL FIRE is an example of a one way variable. It can only deviate positively. The +1 value means that the event EXTERNAL FIRE has occurred. If we were developing the event

T3(-1), we could not trace this to EXTERNAL FIRE (-1) as a local cause. EXTERNAL FIRE (-1) has no physical meaning. T3 is on a negative feedback loop which should be able to handle the +1 deviation. Hence, use the negative feedback operator, and store T3(+1) for later consistency checks and get the following tree.

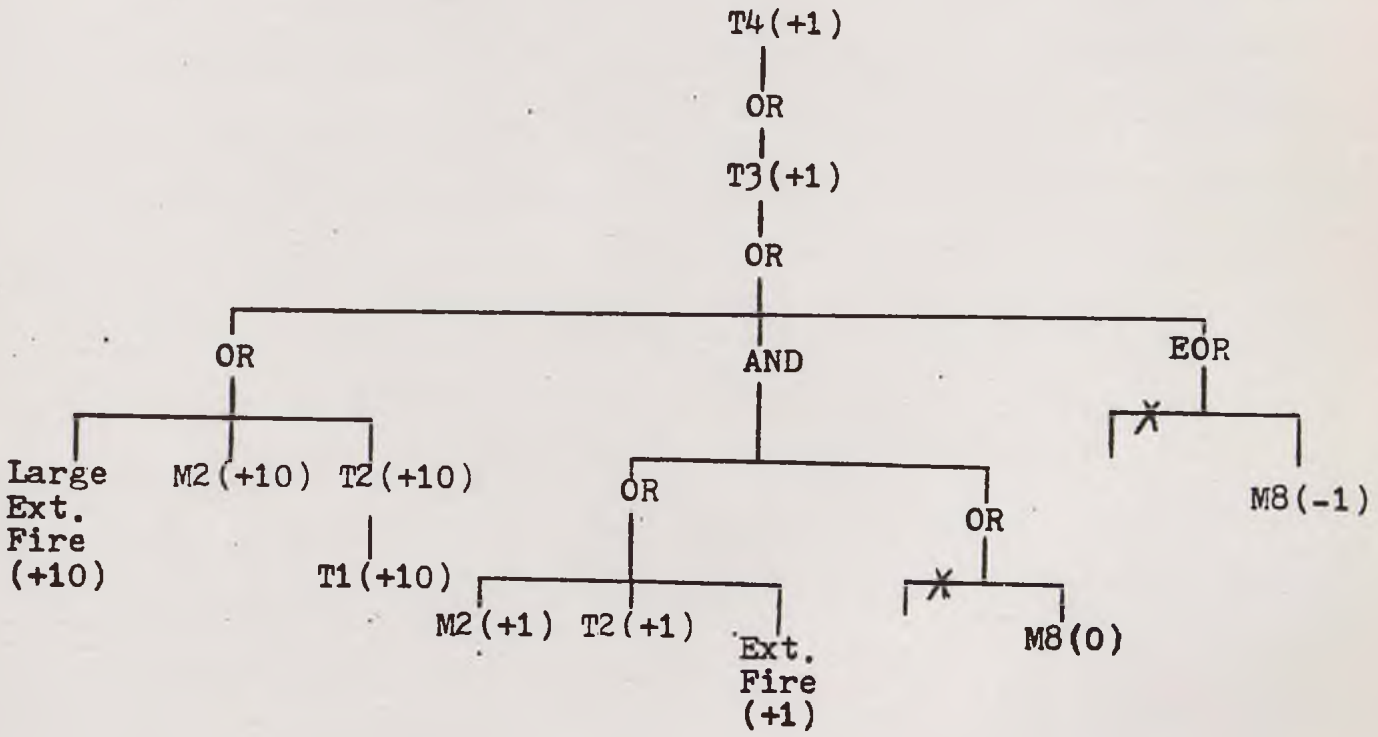


Figure 27: Partial fault tree for the HNO3 process

The X's on the tree indicate that no events are on these branches. The OR gate on the extreme left is a grouping of those large disturbances which enter from off the loop. These disturbances pass directly through the loop. The AND gate in the middle of the tree contains external

disturbances entering the loop as one feed anded with a failure of the negative feedback loop to cancel them out. The external disturbances are grouped under an OR gate. Note that they all have 1 values and hence should be handled by the loop. Failures of the negative feedback loop to cancel disturbances are shown under the other OR gate. One feed to this gate is M8(Ø) which means that M8 did not change due to an inactivation further back on the loop. The other feeds are any local inactivations ie. zero gain edges on the feedback loop. In this case no zero edges exist between M8 and T3 so there are no other feeds. The EOR gate on the right of the tree contains conditions under which either the loop causes the disturbance or passes a disturbance which enters further back on the loop. M8(-1) is an indication that the problem is further back on the loop. The other feed to the EOR gate contains any local events which reverse the polarity of the loop. Events of this type make the loop a positive feedback loop which will drive itself to an extreme due to noise in the system. The EOR gate is necessary because if both events occur, a cancelling signal will be sent forward through the loop. No edge which inverts the loop exists between M8 and T3 so this feed is empty. Note that T3 is also on a negative feedforward loop. Remember, however, that the negative feedforward operator is not invoked until the start of the loop is reached. In this case the start is the event PUMP SHUTDOWN.

Next choose T2(+1) as an undeveloped variable. The

order in which variables are developed makes no difference in the final result. T2 has only one input, that being T1 with a gain of +1. The local cause is therefore T1(+1) and since this variable is on no negative feedback or negative feedforward loops, the appropriate operator is the OR gate. Similar reasoning may be used to determine that the appropriate structure for T2(+1Ø) is an OR gate with T1(+1Ø) as its input. The tree now has the following form.

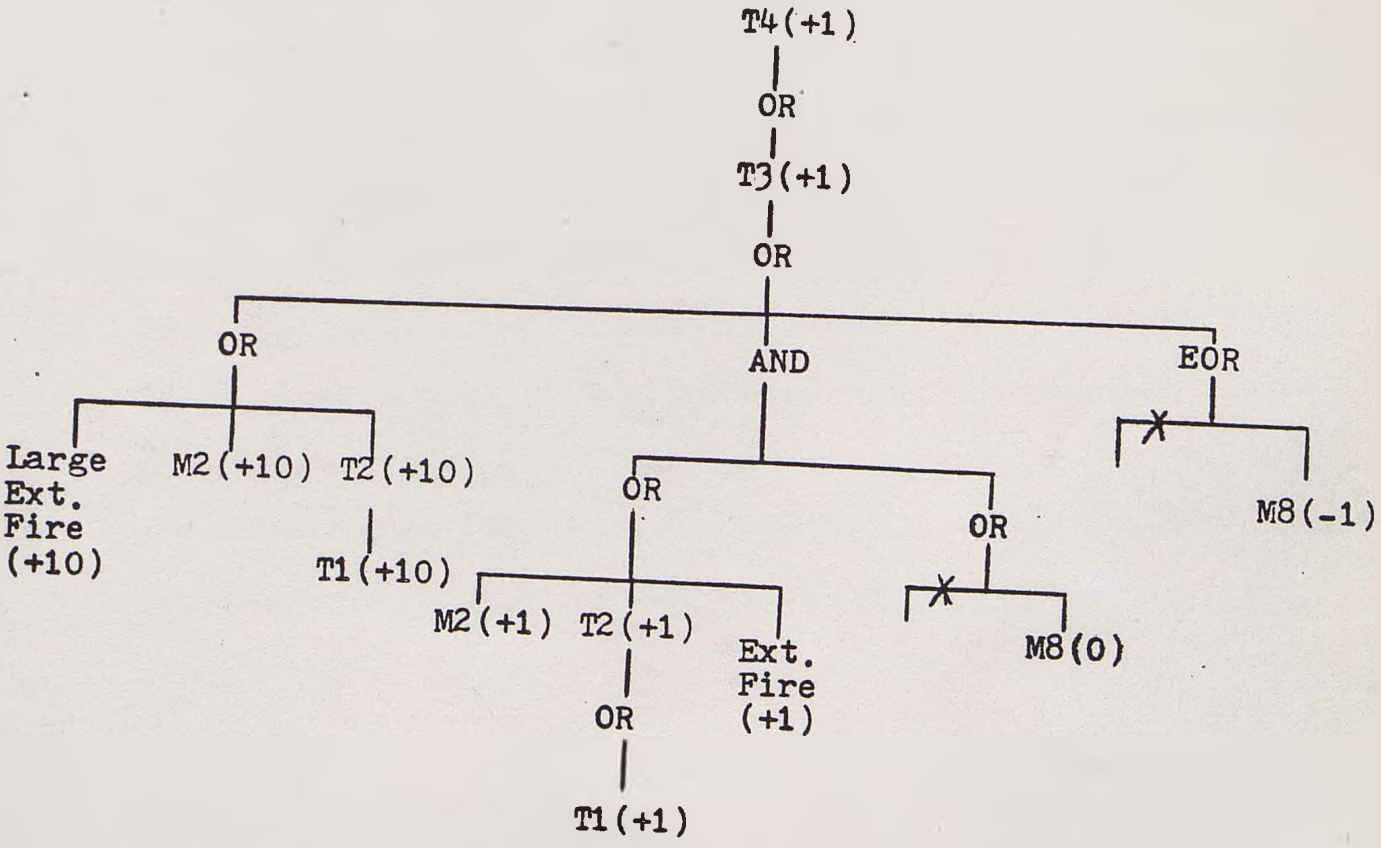


Figure 28: Partial fault tree for the HNO3 process

Now develop M2(+1). Local causes are P1(+1) and P11(-.1).

The value .1, however, is not allowed for the magnitude of a disturbance. Only Ø, 1, and 1Ø are allowed values. The 1Ø gains between P11 and M2 imply that if P2 deviates, a large change is seen in M2. P11(+1) with the condition REVERSED VALVE ACTION is not considered here since the negative feedforward operator, when applied, will handle all conditional edges on the loop. If M2 were not on the negative feedforward loop, then P11 with the REVERSED VALVE ACTION condition would have to be considered. We delete all causes with .1 magnitude which results in P1(+1) as the only local cause. M2 is on the negative feedforward loop but it is not at the start so again employ the OR gate with P1(+1) as the sole input. Next develop M2(+1Ø). Local causes are P1(+1Ø) and P11(-1). P11(+1) with REVERSED VALVE ACTION is not considered for reasons stated above. Again employ an OR gate with feeds P1(+1Ø) and P11(-1). Now expand P11(-1). The only local cause is PUMP SHUTDOWN (-1). PUMP SHUTDOWN, like EXTERNAL FIRE, however, is an event which only deviates positively. PUMP SHUTDOWN (-1) has no meaning so delete it and note that P11(-1) has no local causes. The tree developed to this point appears on the next page.

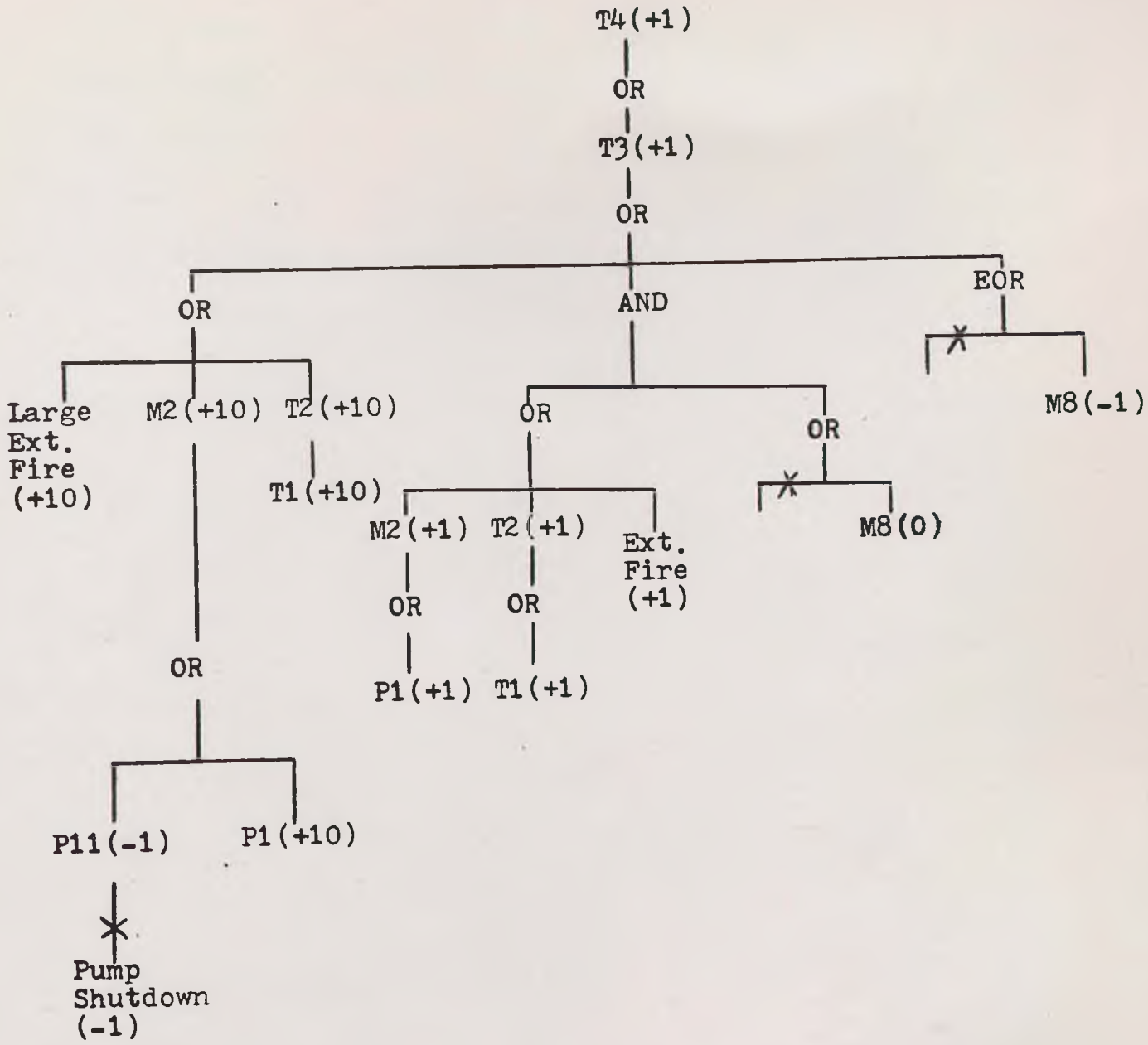


Figure 29: Partial fault tree for the HNO3 process

Next proceed with M8(-1). The local causes are P9(-1) and P7(-1). Neither of these violates the consistency requirement T3(+1). M8 is on a negative feedback loop and negative feedforward loop. We have not reached the start of the negative feedforward loop, however, so employ the negative feedback operator. P9(-10) is the large

disturbance off the loop while P9(-1) is the normal disturbance. No local inactivating edges exist between M8 and P7 but a local inversion edge, REVERSED VALVE ACTION, is present. Using the negative feedback operator and these events, arrive at the following tree.

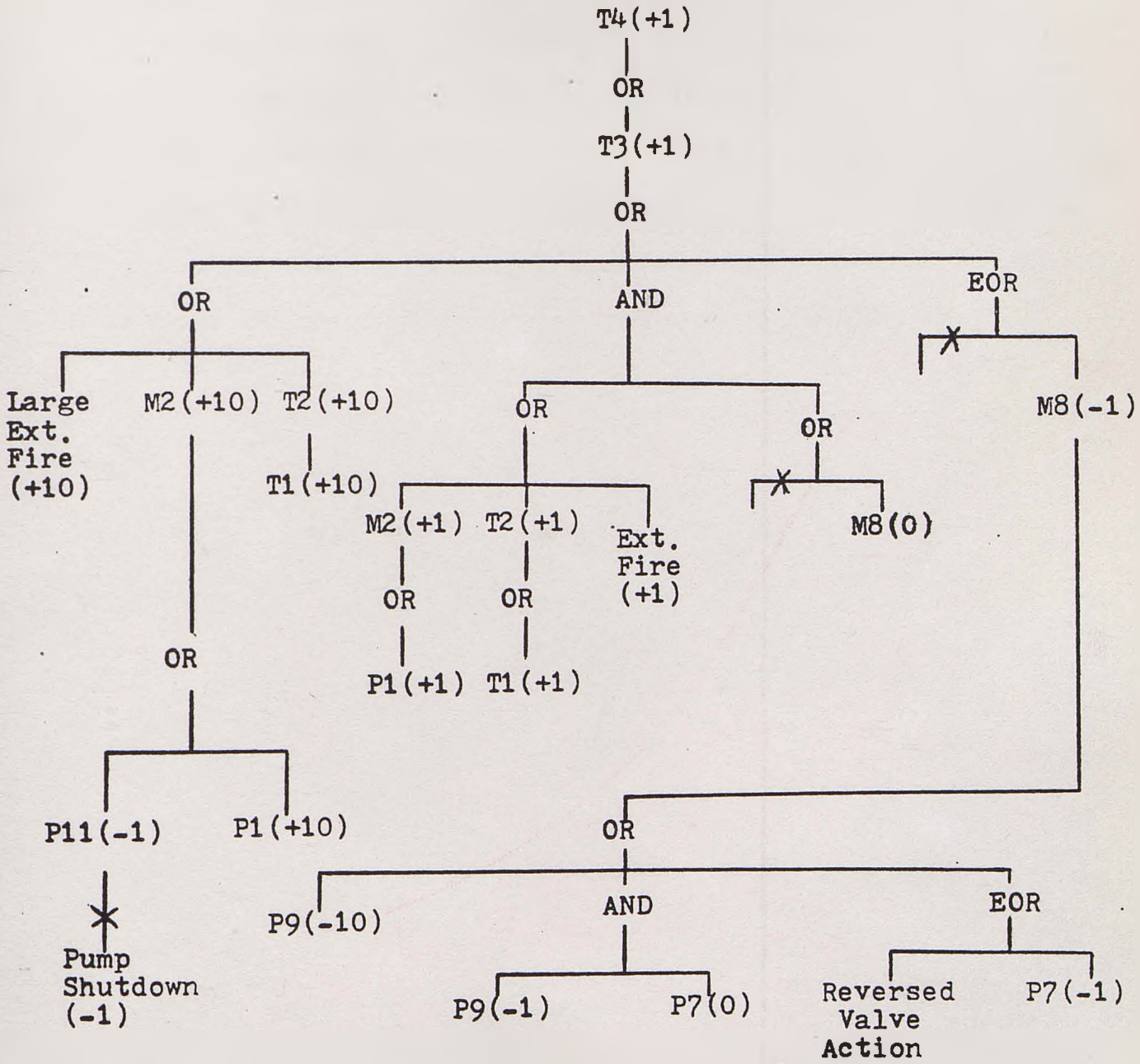


Figure 30: Partial fault tree for the HNO3 process

Now do P7(-1). Local causes are LOW AIR PRESSURE (+1),

EXTERNAL FIRE AT LOCATION 4 (-1), and P6(-1). Delete EXTERNAL FIRE AT LOCATION 4 (-1) since this event can only deviate positively. P7 is on the negative feedback loop so use the negative feedback loop operator. The large deviation off the loop is LOW AIR PRESSURE (+10), ie. a complete loss of instrument air. The normal disturbance off the loop is LOW AIR PRESSURE (+1). Both an inactivation edge (CONTROLLER BROKEN) and an inversion edge (CONTROLLER ACTION REVERSED) exist. Next develop P6(-1). Only one local cause exists and this is T3(-1). T3(-1), however, violates the consistency requirement T3(+1). Hence no causes exist and the tree now has the following form.

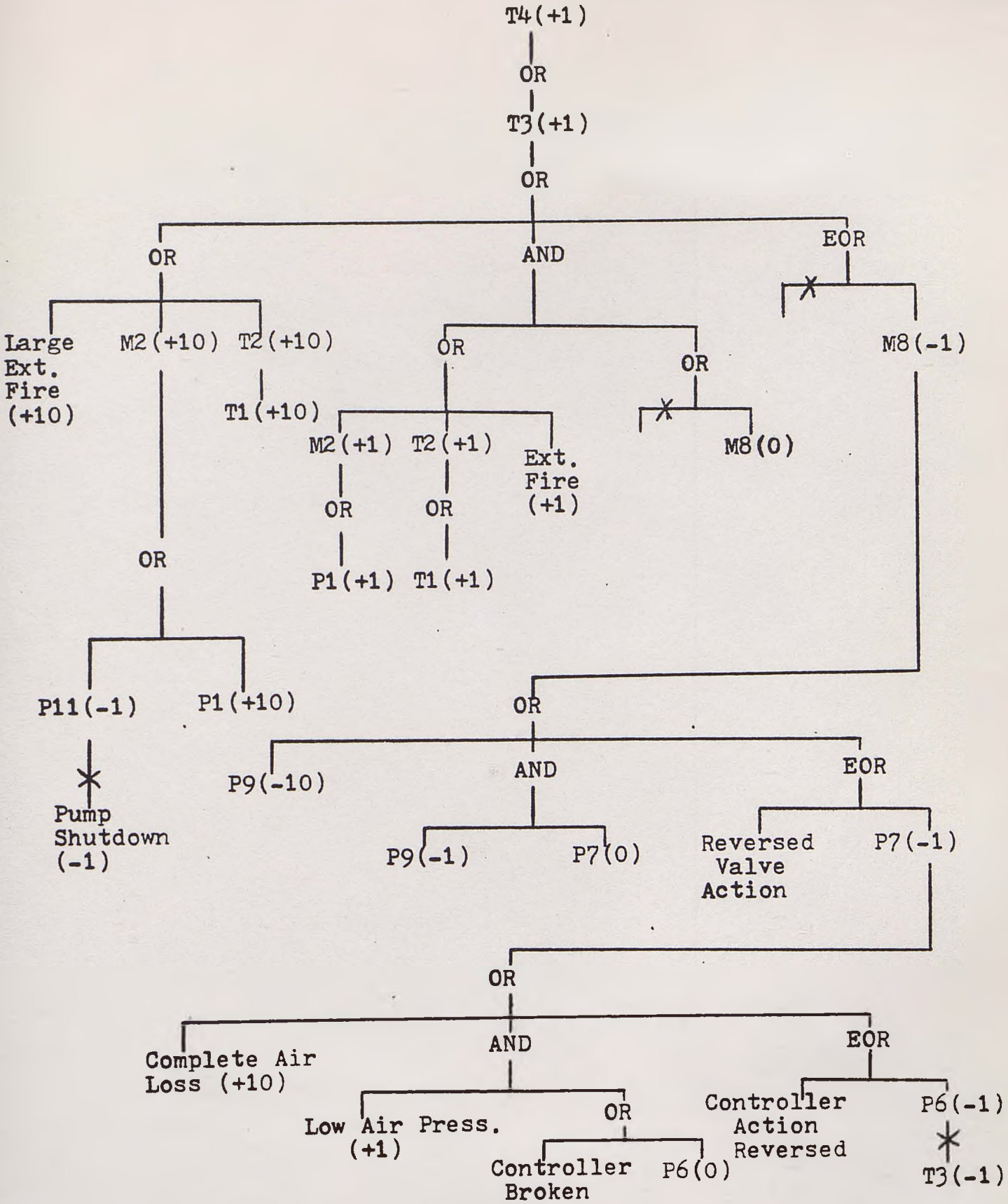


Figure 31: Partial fault tree for the HNO3 process

Now develop the events with zero values. Note that all

of them are variables which are on the negative feedback loop. They are variables which should be changing under the condition T3(+1), but are not. In order to develop these, we proceed stepwise back along the feedback loop looking for zero edges. In the case of M8(Ø), no zero edge exists so the only cause is P7(Ø). The appropriate operator is an OR gate since either local inactivators or the next variable back on the loop not changing, or both will result in no deviation in the variable currently under development. The tree follows.

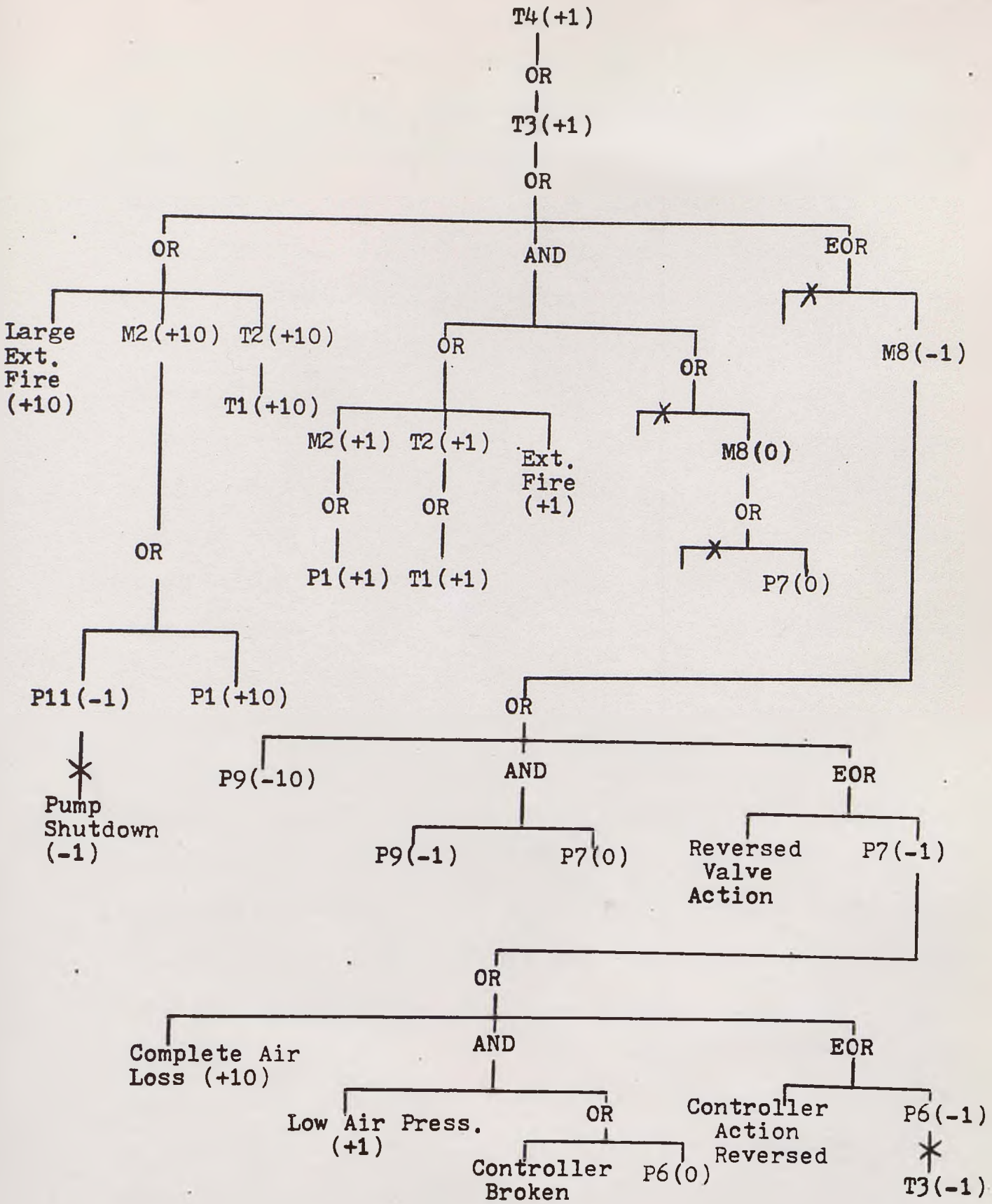


Figure 32: Partial fault tree for the HNO3 process

Note that P7(Ø) appears twice in the tree. If the

consistency requirements are the same, we may expand P7(Ø) once and place the same sub-tree under both occurences of P7(Ø). One local inactivator exists (CONTROLLER BROKEN) so use an OR gate with this event as one input and P6(Ø) as the other. Now do P6(Ø). Like P7(Ø), this also appears twice so place the same development under both occurrences. SENSOR BROKEN is a local inactivator so use an OR gate with this event and T3(Ø) as inputs. T3(Ø), however, must be delete since it violates the T3(+1) consistency requirement. We have now completed the development of the entire negative feedback loop and the tree is shown in figure 33. P9(-1) needs further development. Local causes are P1Ø(-1) and PUMP SHUTDOWN (+.1). PUMP SHUTDOWN (+.1) is not allowed so P1Ø(-1) is the only cause we need to consider. Although we have reached the start of the negative feedforward loop, the starting variable itself has been deleted. P9 is not on the negative feedback loop so use the OR operator with P1Ø(-1) as its input. Next do P9(-1Ø). Local causes are P1Ø(-1Ø) and PUMP SHUTDOWN (+1). Since PUMP SHUTDOWN starts the negative feedforward loop, employ the negative feedforward operator to arrive at the tree shown in figure 34. The events M2(+1Ø) and M2(Ø) are to be developed only along the other branch of the negative feedforward loop. M2(+1Ø) is an indication that an inversion exists along this path while M2(Ø) indicates inactivation along the path. Both events will carry the consistency requirement PUMP SHUTDOWN (+1). Developing M2(+1Ø), we arrive at two local causes these

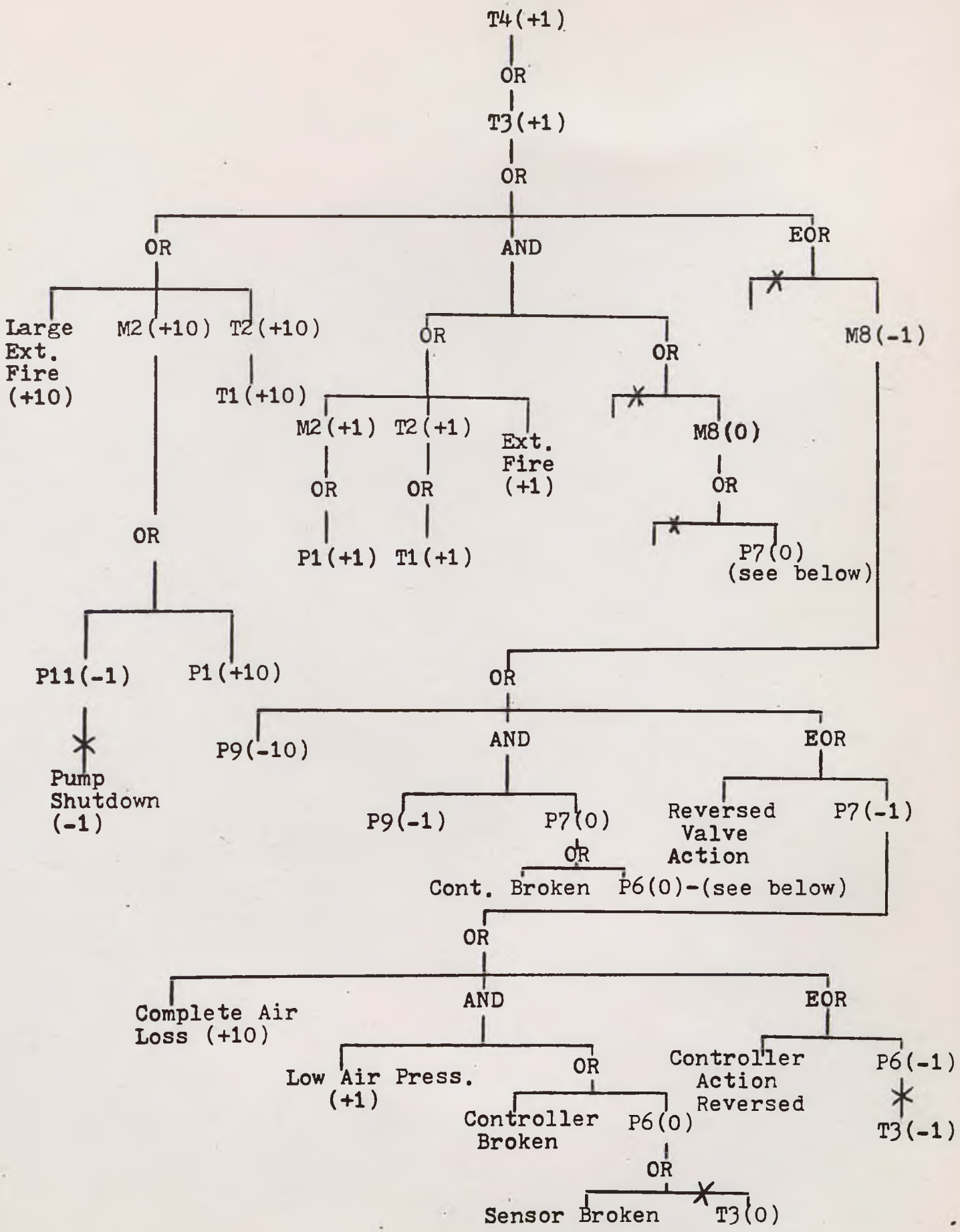


Figure 33: Partial fault tree for the HNO3 process

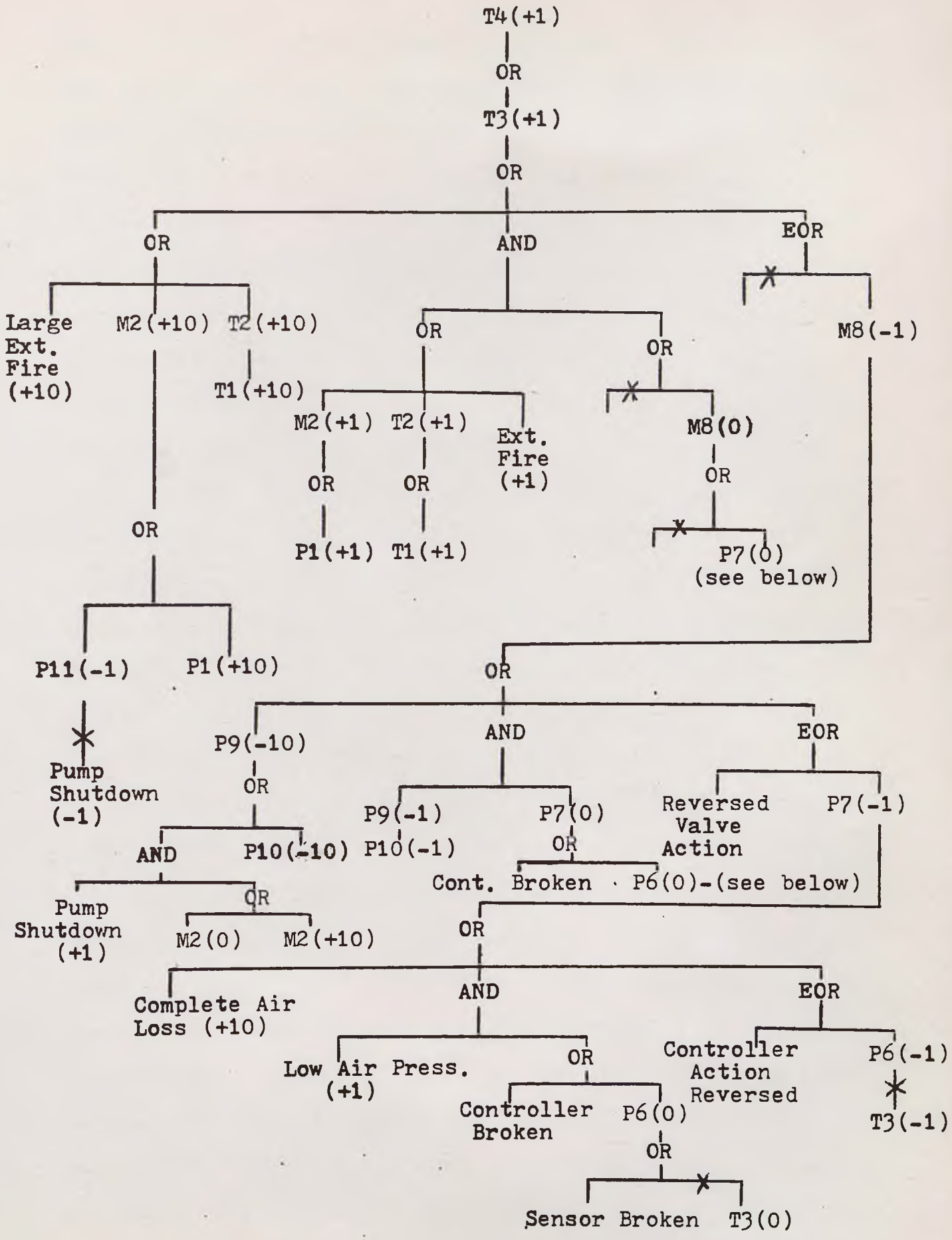


Figure 34: Partial fault tree for the HNO3 process

being REVERSED VALVE ACTION and P11(-1). Note that we do not have to state the condition P11(+1) with REVERSED VALVE ACTION since this event is normally true given PUMP SHUTDOWN (+1). An EOR gate is the proper operator since, if both events occur, a cancelling signal will be sent forward. Next do P11(-1) and determine that the only cause is PUMP SHUTDOWN (-1). This is both illegal and inconsistent so delete it. The tree at this point is shown in figure 35. Finally we must develop M2(Ø). We do this in exactly the same way as we developed zeros along the negative feedback loop. M2(Ø) has no local inactivators so use an OR gate with P11(Ø) as its input. P11 has one inactivator (LINE 11 PLUGGED) so use an OR gate with it and PUMP SHUTDOWN (Ø) as feeds. Delete PUMP SHUTDOWN (Ø), however, since it violates the PUMP SHUTDOWN (+1) consistency requirement. The fault tree is now complete and is shown in figure 36. Gates with no feeds and single feeds have been removed.

This tree was developed in a rigorous fashion. Several useful short cuts could have been used. First, when looking for inactivators along either negative feedback or negative feedforward loops, it is not necessary to proceed stepwise along the loop. One simply needs to use an OR gate with the appropriate edges as inputs. The same thing may be done when looking for reversals along a negative feedforward loop except that an EOR gate must be used as the operator. Second, duplicate events need only be developed once. One must be careful, however, that the events carry the same

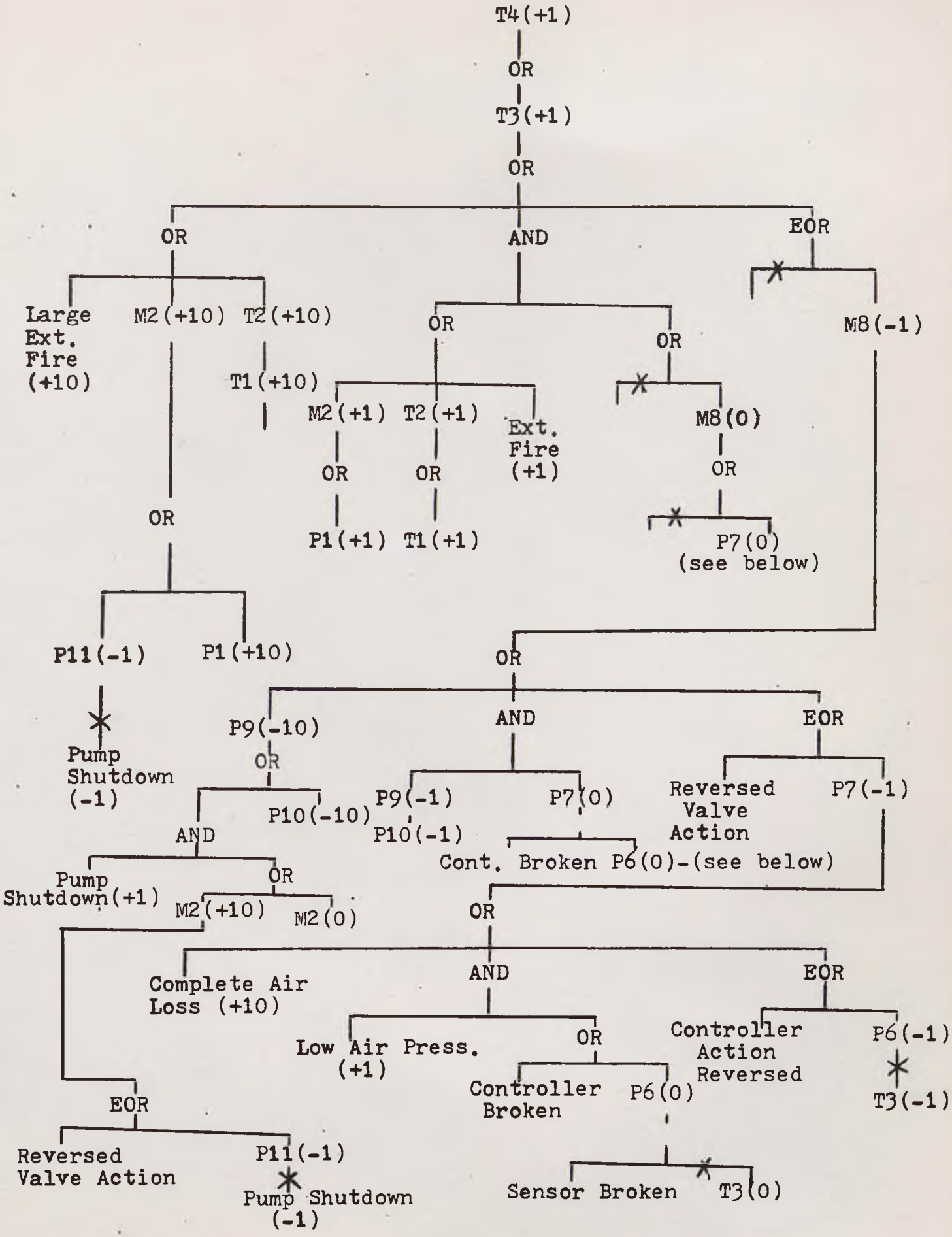


Figure 35: Partial fault tree for the HNO3 process

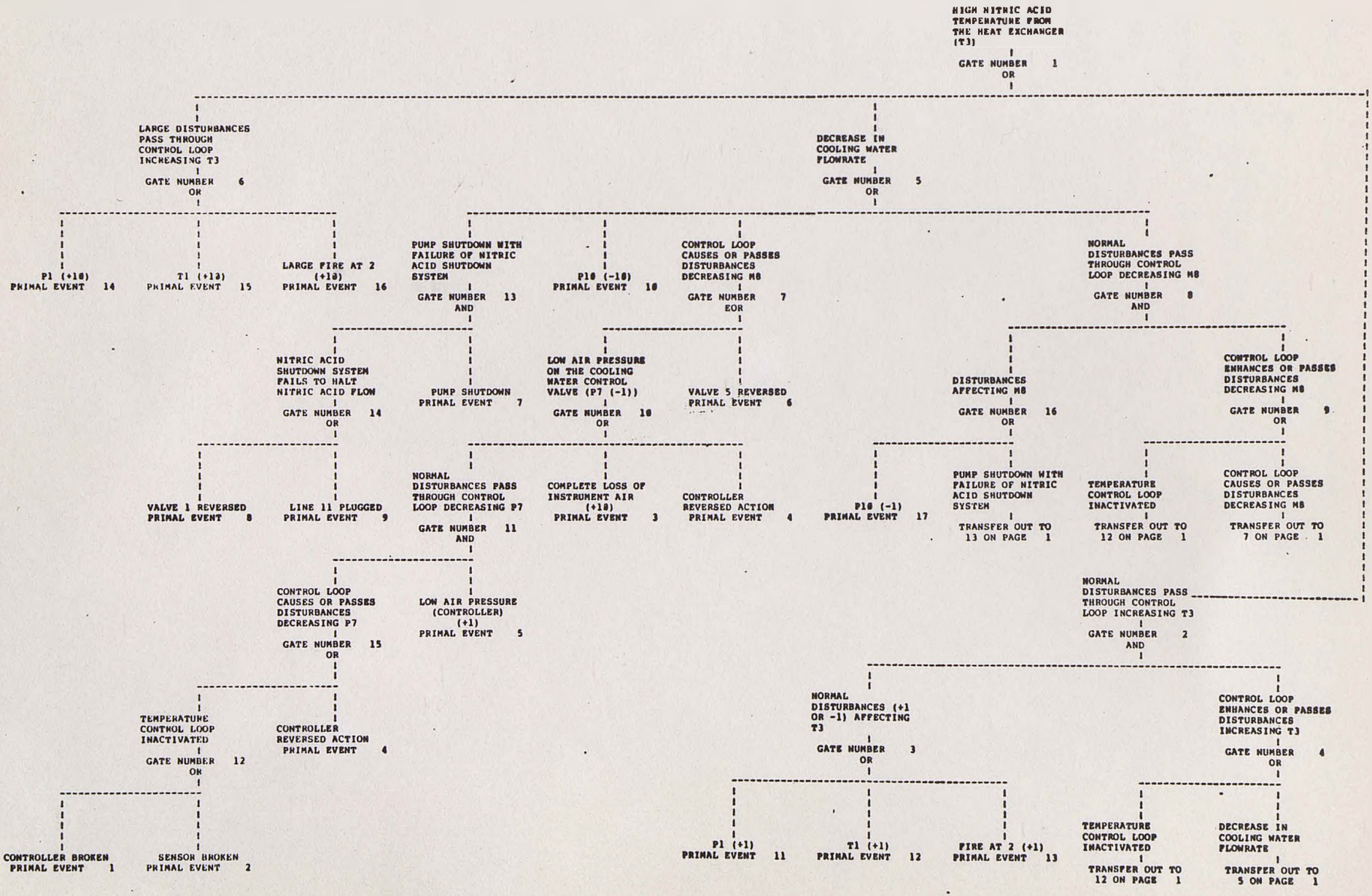


Figure 36: Complete fault tree for the HNO3 process

consistency requirements with them. This was true for P7(Ø) and P6(Ø) so the development was only done once. It was not true for M2(+1Ø) since in one case we had the consistency requirement PUMP SHUTDOWN (+1) and in the other case we did not. Because of this, M2(+1Ø) was developed twice with different sub-trees.

We have now gone completely through the process of constructing a fault tree for a specific process. All of the basic concepts of modelling and fault tree construction have been demonstrated. In the following sections, we shall present some extensions on the basic ideas which will prove useful in analyzing other systems.

MODELLING EXTENSIONS

The nitric acid process illustrated how to model process components. In addition to models for the equipment, we also need digraph models for the human operator. Fortunately, operating procedures and their associated failures can be modelled using digraphs. Consider the flowsheet shown in figure 37. An operating procedure for this process is that if the high pressure alarm comes on, the operator is to move the switch for valve V1 to the closed position. This should stop the feed and halt the pressure rise. Using digraph models for the reactor, valve, switch, pressure alarm, and human operator, construct the system digraph for the event REACTOR PRESSURE (+1Ø) shown in figure 38. The connections between HPA and V1 SWITCH were developed using a model of the human operator given the stated operating procedure. Note that the operator appears on the digraph in a position similar to where a continuous PID controller might appear. This is because the operator functions similar to the way a controller would work. He senses a signal - the high pressure alarm. Then an action is computed using the operating procedure. Finally an action is taken - the V1 switch is actuated. As shown on the digraph, failure mdoes for the operator can also be easily captured. The ability to capture operator action on the digraph is a valuable feature of this type of modelling. It allows the analysis of possible failures in the operating procedure as well as failures in the equipment. This is

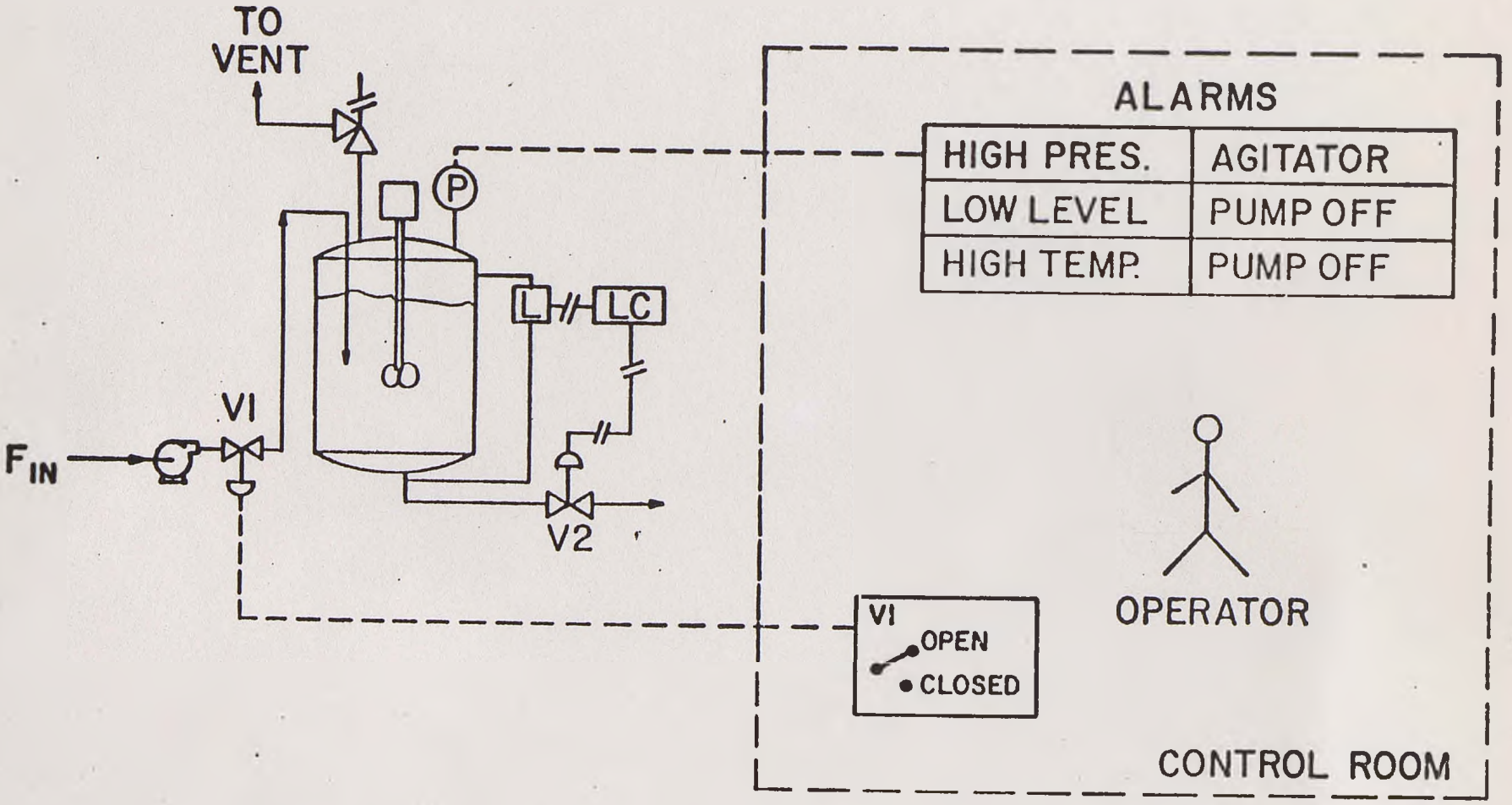


Figure 37. Operator-Process Flowsheet

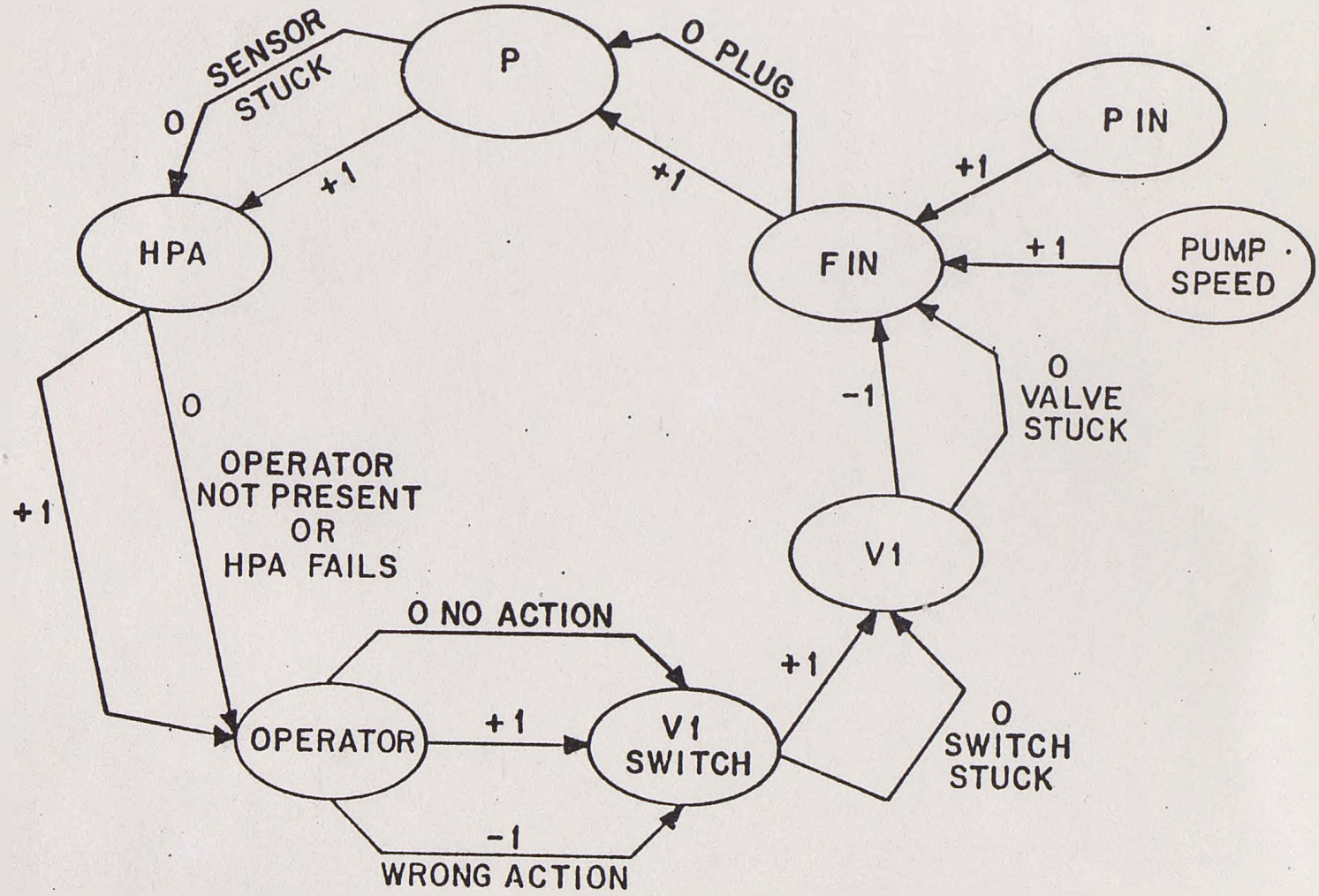


Figure 38 Operator-Process Digraph

important since the operator is a critical component in many chemical processes.

The systems discussed up to this point have been continuous in nature. Other important types of systems are those which are sequential. Systems of this type have the characteristic that the process configuration changes with time. Probably the best examples of these systems are batch processes. Sequential systems present special problems in analysis since events which have occurred in previous time periods often determine which failures are most important in the current period. In order to construct fault trees for sequential systems, we must first determine if digraph models can capture sequential behavior.

Consider the system shown in figure 39. This is a system designed to dry air by passing it through two alumina beds. One bed is used for drying while the other is regenerating. The system is sequential since, after a certain period of time, the two 4 way valves change position causing the beds to change service. Consider a model for a 4 way valve. The digraph model for flows through a 4 way valve is shown in figure 40. Note that the flow from stream 4 (F4) may be affected by either the flow in stream 1 or that in stream 2 depending on valve position. We are able to capture this by drawing appropriate edges and labelling them with the conditions under which they apply. Recall that the models we developed previously always had one edge between nodes which had no condition on it. This edge

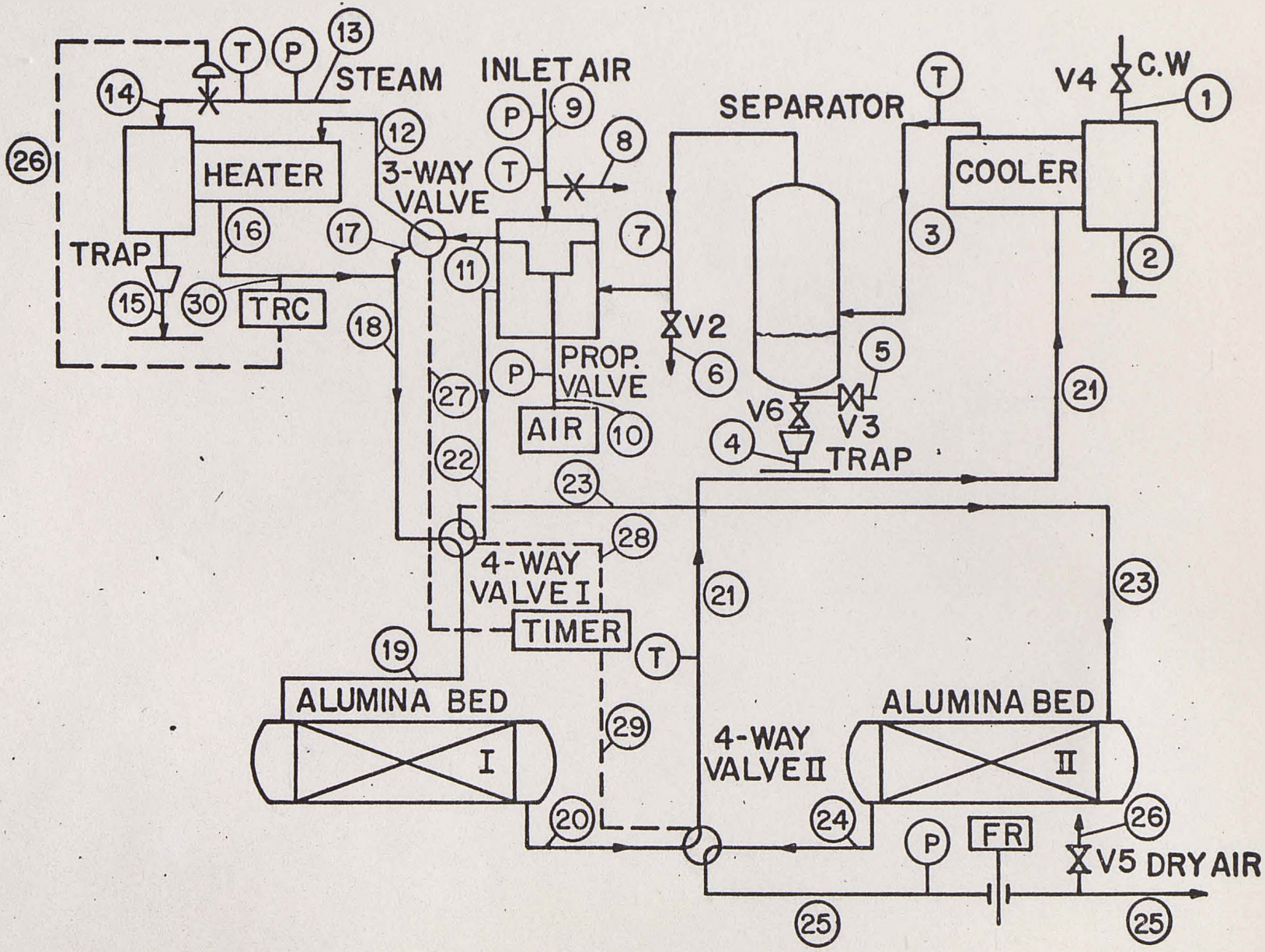
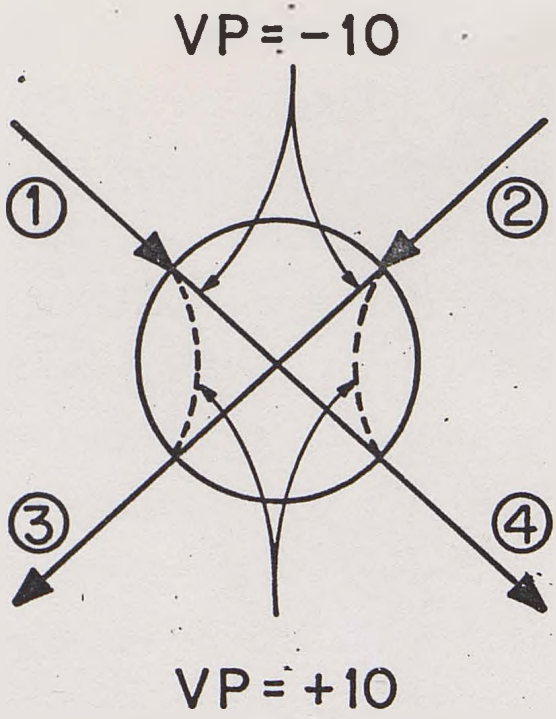


Figure 39: FLOW DIAGRAM OF UTILITY AIR DRYING UNIT



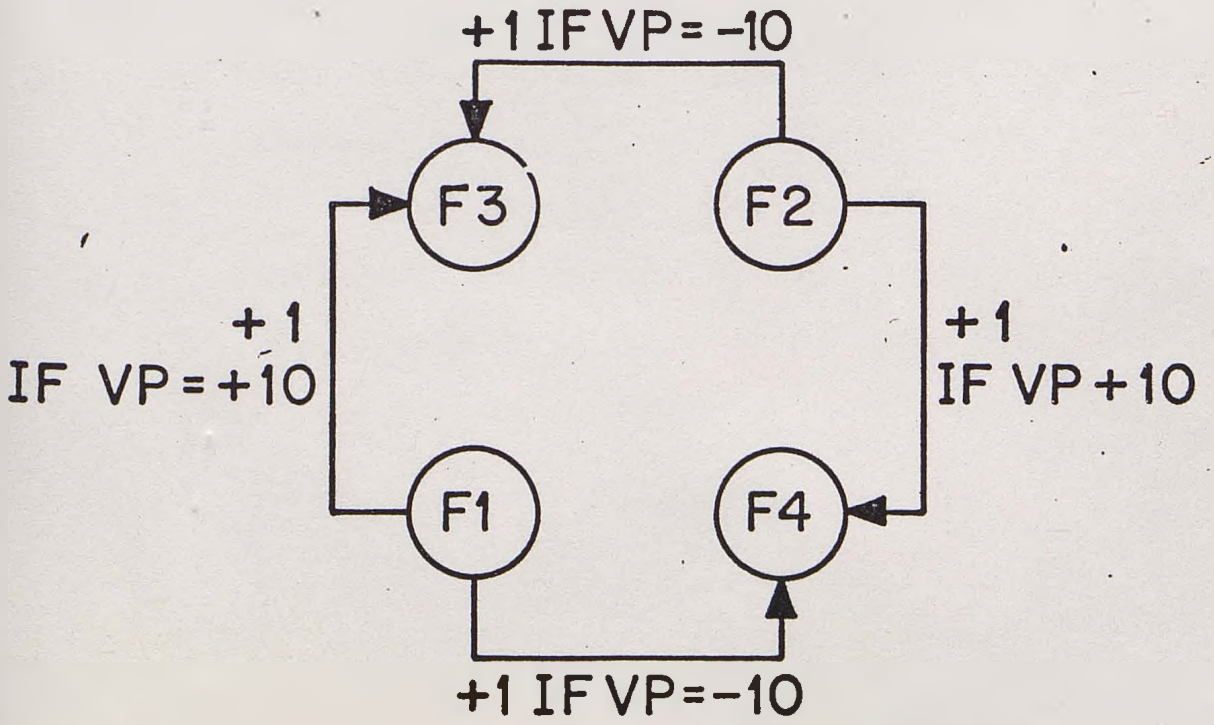


Figure 40: 4 Way Valve Model

represented the normal operating state. With sequential devices, however, the normal state is a function of some other variable. Hence sequential models will always have conditions on their edges which state the conditions under which those edges apply. The condition itself may be traced further back in the digraph. For instance, since the 4 way valve position is controlled by a timer, we would trace it further using a digraph model of the timer. The timer model for the dryer process is shown in figure 41. Once the entire digraph has been developed, we may then proceed with the synthesis of the fault tree using the operators previously discussed. Conditions which are time dependent will naturally appear on the fault tree since they are already on the digraph. The top of the fault tree for the event HIGH WATER CONCENTRATION IN STREAM 25 is shown in figure 42. Note that some failures are time dependent while others are not. A more complete discussion on sequential fault tree analysis of this system has appeared in the literature [11]. A copy of the article is in appendix A.

Having presented some extensions on digraph models, we shall next present some extensions on the operators used to construct fault trees. These extensions will considerably enlarge the types of systems which can be handled.

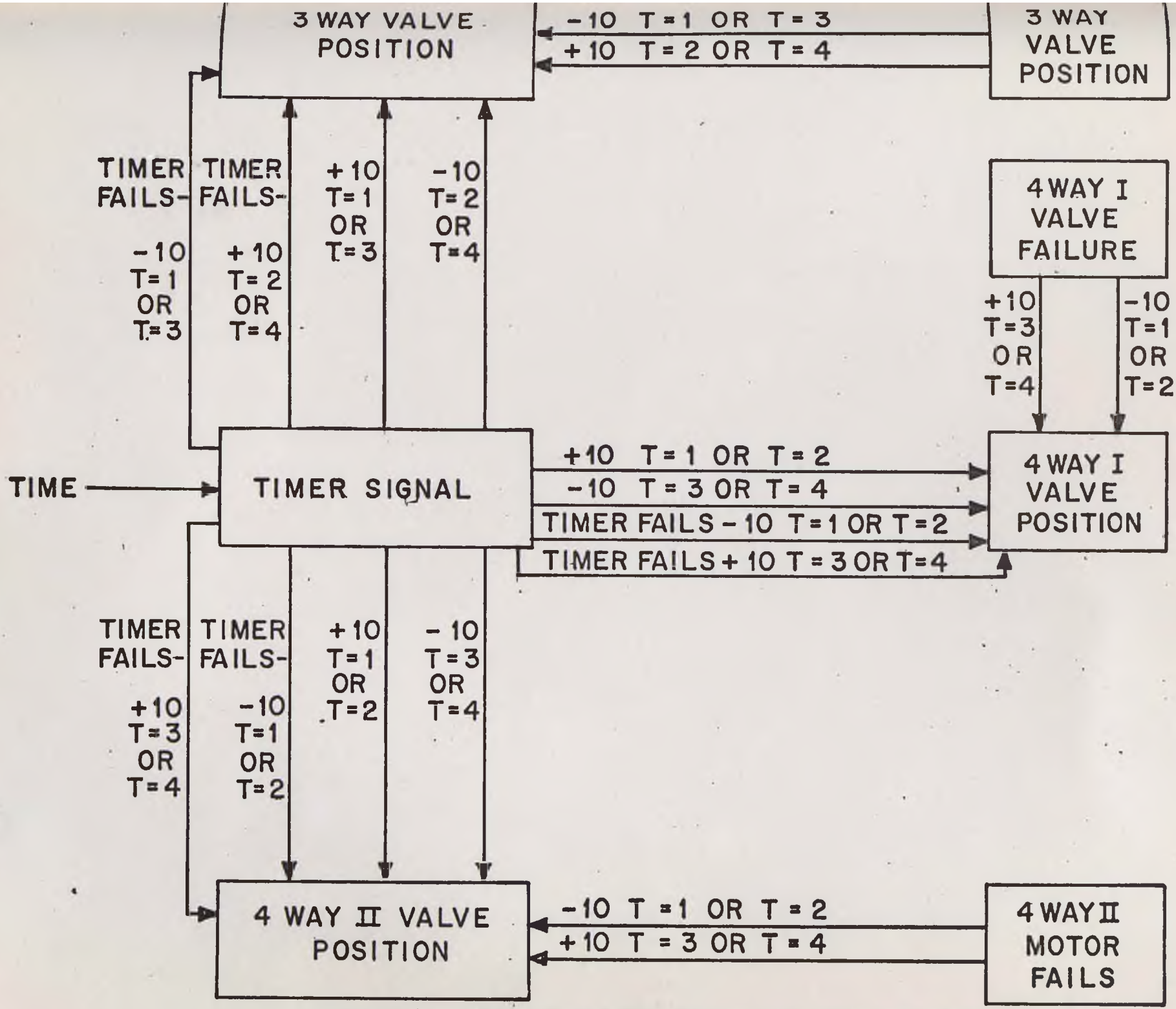


Figure 41: Timer Model for Sequential Dryer Process

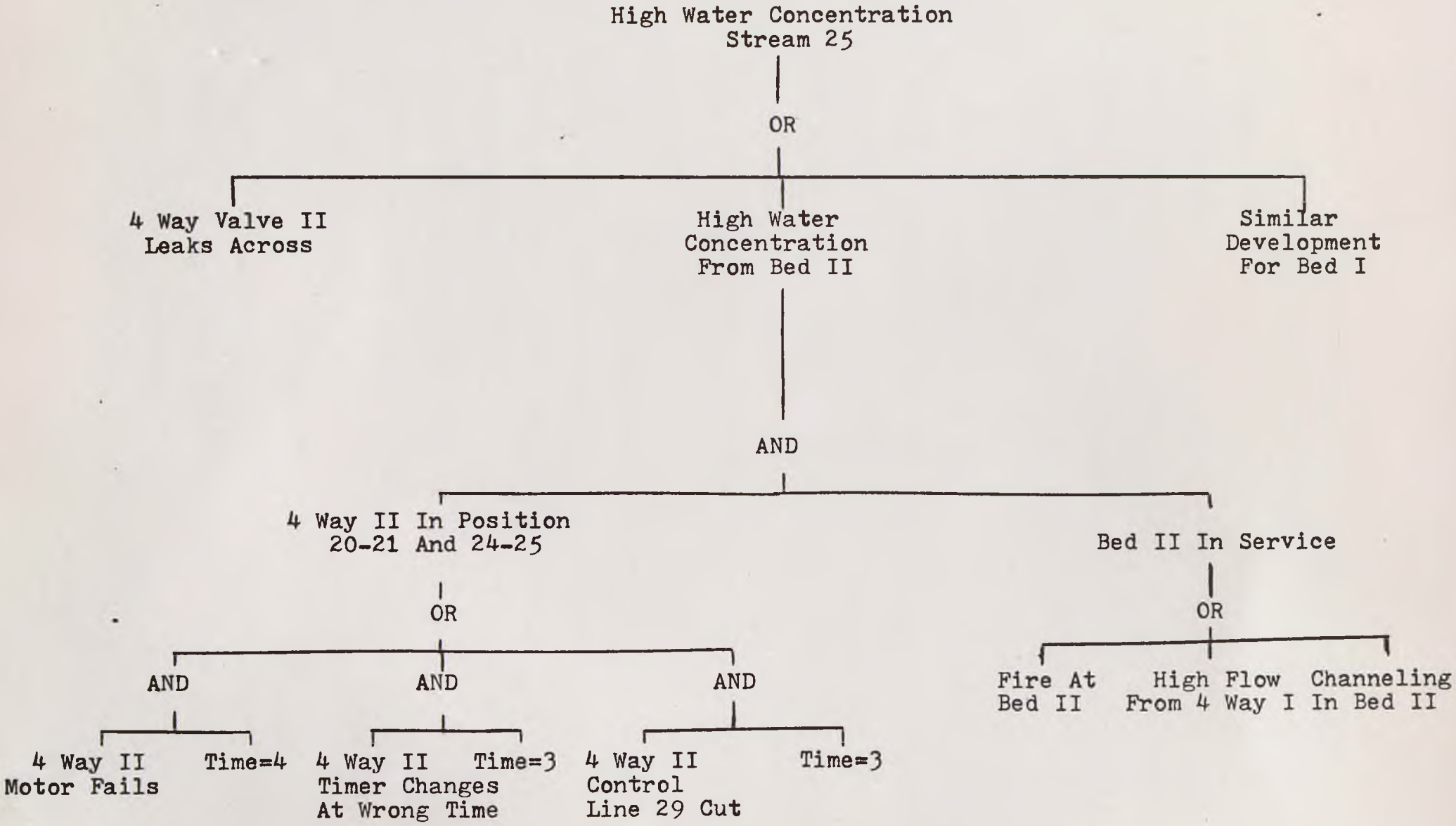


Figure 42: Top of Fault Tree for the Sequential Dryer Process

EXTENSIONS OF THE SYNTHESIS OPERATORS

What happens if a variable is on two negative feedback loops? For a single negative feedback loop, there were three ways of getting a disturbance through the loop. These were first to send a disturbance into the loop which it was not designed to handle, second to send a disturbance which the loop should handle and then inactivate the loop, or third have the loop itself cause the disturbance. When two negative feedback loops are involved, large disturbances will still go through both loops so that a portion of the operator remains unchanged. Normal disturbances, however, must also get through both loops. Assuming that either loop is capable of removing the disturbance, we must inactivate the negative feedback system. This can be done by either inactivating both loops or by having the loops fight one another. Hence, three possibilities exist, inactivate both loop 1 and loop 2, reverse loop 1 and keep loop 2 normal, or reverse loop 2 and keep loop 1 normal. Of course, the events which are normally true are not developed. The loops could also cause a disturbance in three possible ways. They are reverse both loops, reverse loop 1 and inactivate loop 2, or reverse loop 2 and inactivate loop 1. The complete fault tree operator is shown in figure 43. Note that this operator assumes both loops to be of equal strength and speed. If the loops differ, then certain parts of the operator will change. For instance, if loop 1 is much faster than loop 2 then in order to get a disturbance

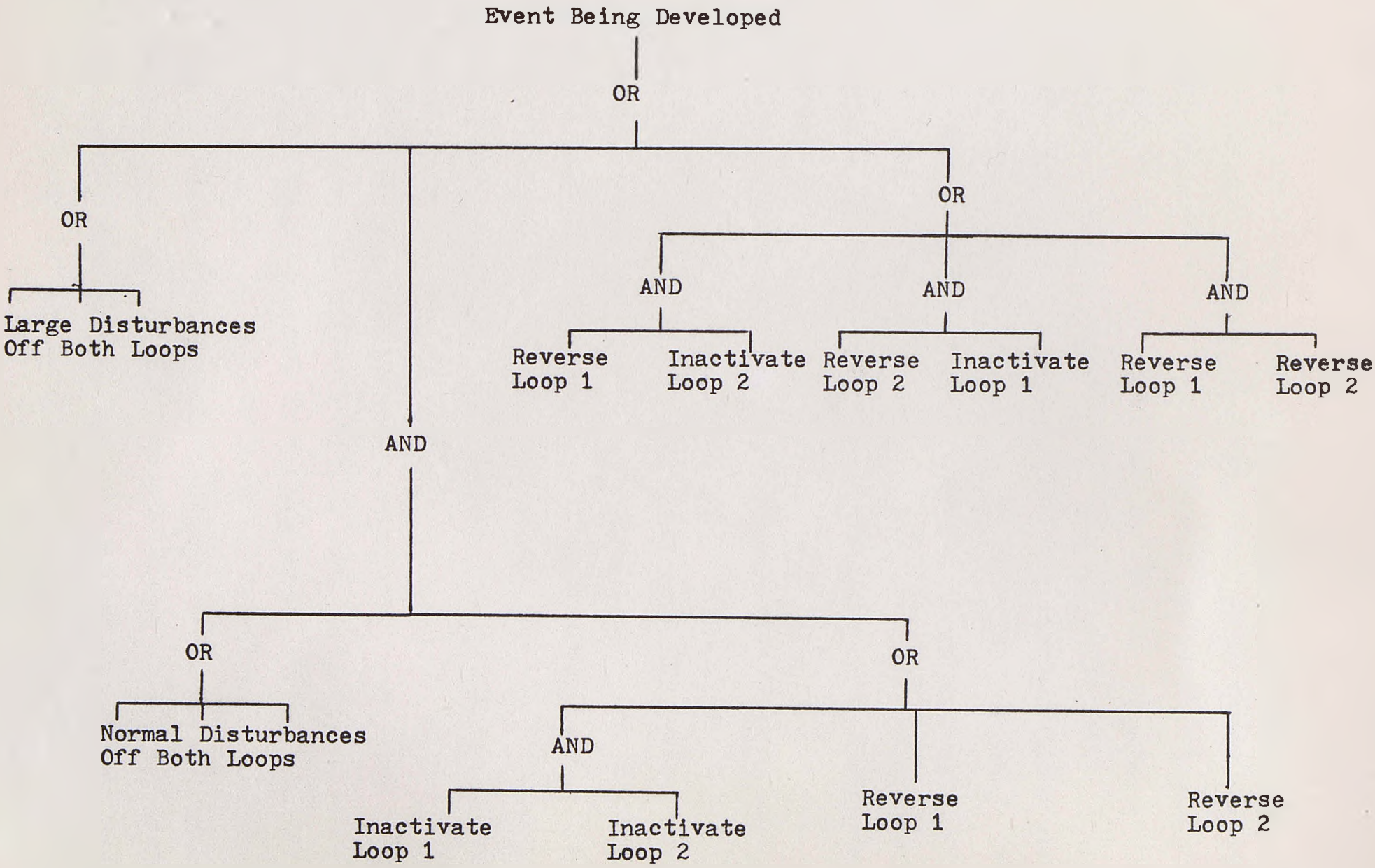
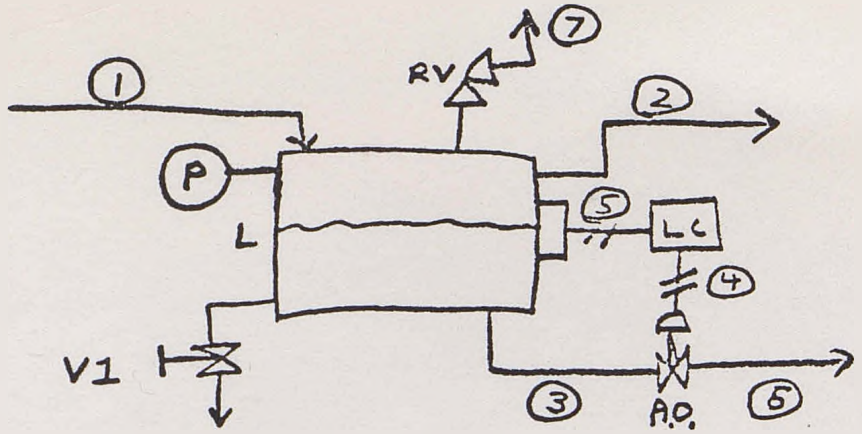


Figure 43: Operator for Two Negative Feedback Loops

through the system, it is only necessary to reverse loop 1. Since loop 2 is slower than loop 1, it is in effect inactive under these conditions. Note that if one loop is assumed to always be inactive, the operator reverts to the single negative feedback loop case. For three or more loops, similar extensions may be made. One simply lists those permutations which produce the desired results. The extensions needed to handle negative feedforward loops with more than two branches are exactly those presented above. They are, of course, applied to the negative feedforward loop operator.

To demonstrate the use of the multiple loop operator, consider to system in figure 44. The digraph shown is developed for the top event TANK PRESSURE (+1Ø). This variable is on two negative feedback loops, both of which should be able to handle this disturbance. One loop corresponds to the relief valve while the other is through the human operator. The procedure modelled is that the operator is to open the dump valve V1 to relieve the pressure if the high pressure alarm comes on. The fault tree is shown in figure 45. Note the application of the multiple loop operator to the top event. This tree also illustrates how to handle loops of different capacities. When developing L(+1Ø), the multiple loop operator is not employed even though L is on two negative feedback loops. The reason for this is that one of the loops, the level control loop, is unable to handle the +1Ø deviation. For


1
RV
7
2
P
L
5
LC
4
V1
3
A.O.
6

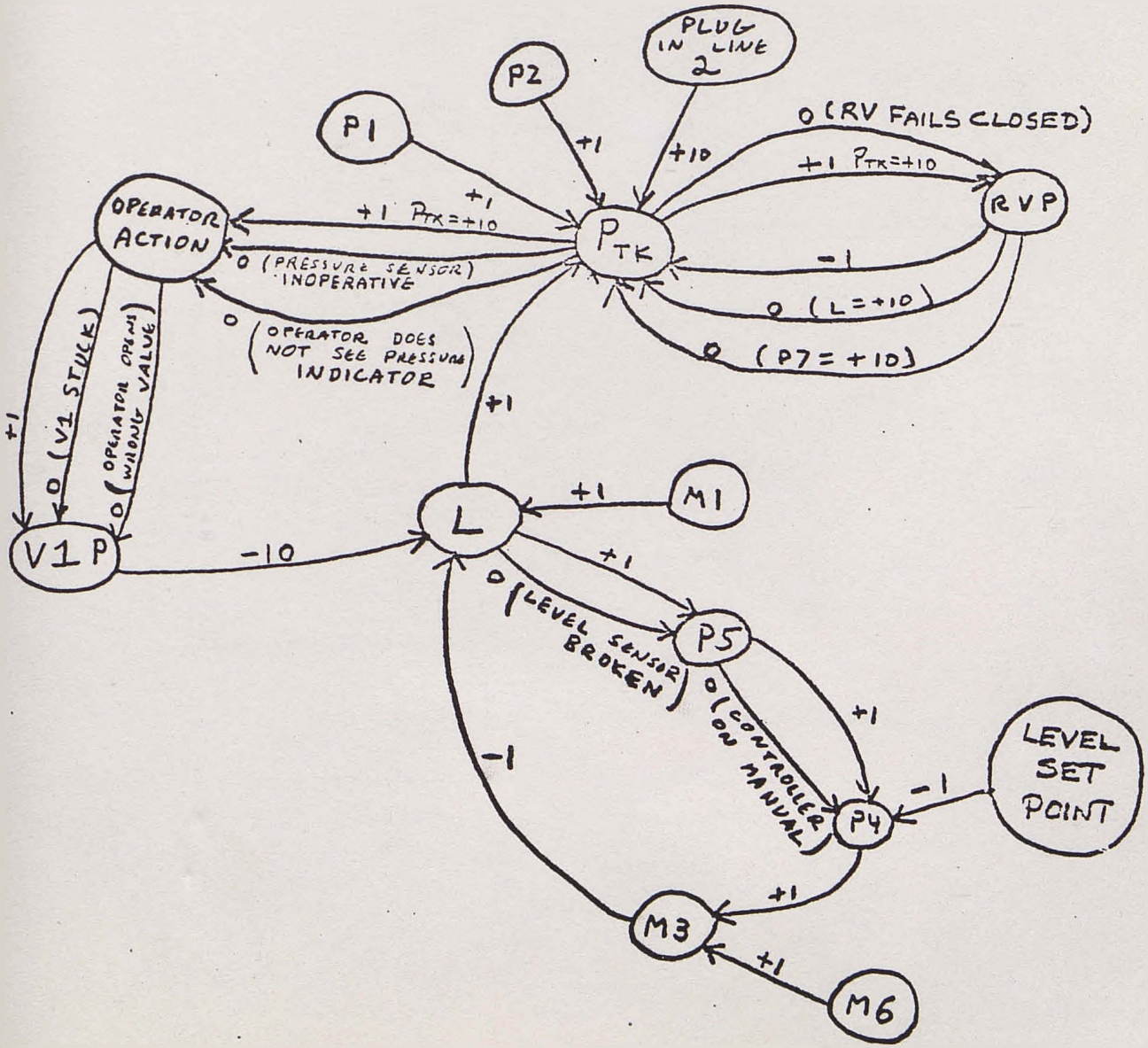

PLUG IN LINE 2
P2
P1
+1
+10
0 (RV FAILS CLOSED)
+1 PTK=+10
RVP
OPERATOR ACTION
+1 PTK=+10
+1
PTK
0 (PRESSURE SENSOR INOPERATIVE
-1
0 (L=+10)
0 (OPERATOR DOES NOT SEE PRESSURE INDICATOR)
0 (P7=+10)
+1
0 (V1 STUCK)
0 (OPERATOR OPENS WRONG VALVE)
+1
M1
L
V1P
-10
+1
0 (LEVEL SENSOR BROKEN)
P5
0 (CONTROLLER ON MANUAL)
+1
LEVEL SET POINT
-1
-1
P4
+1
M3
+1
M6

Figure 44

FAULT TREE

FOR TANK PRESSURIZATION PROBLEM

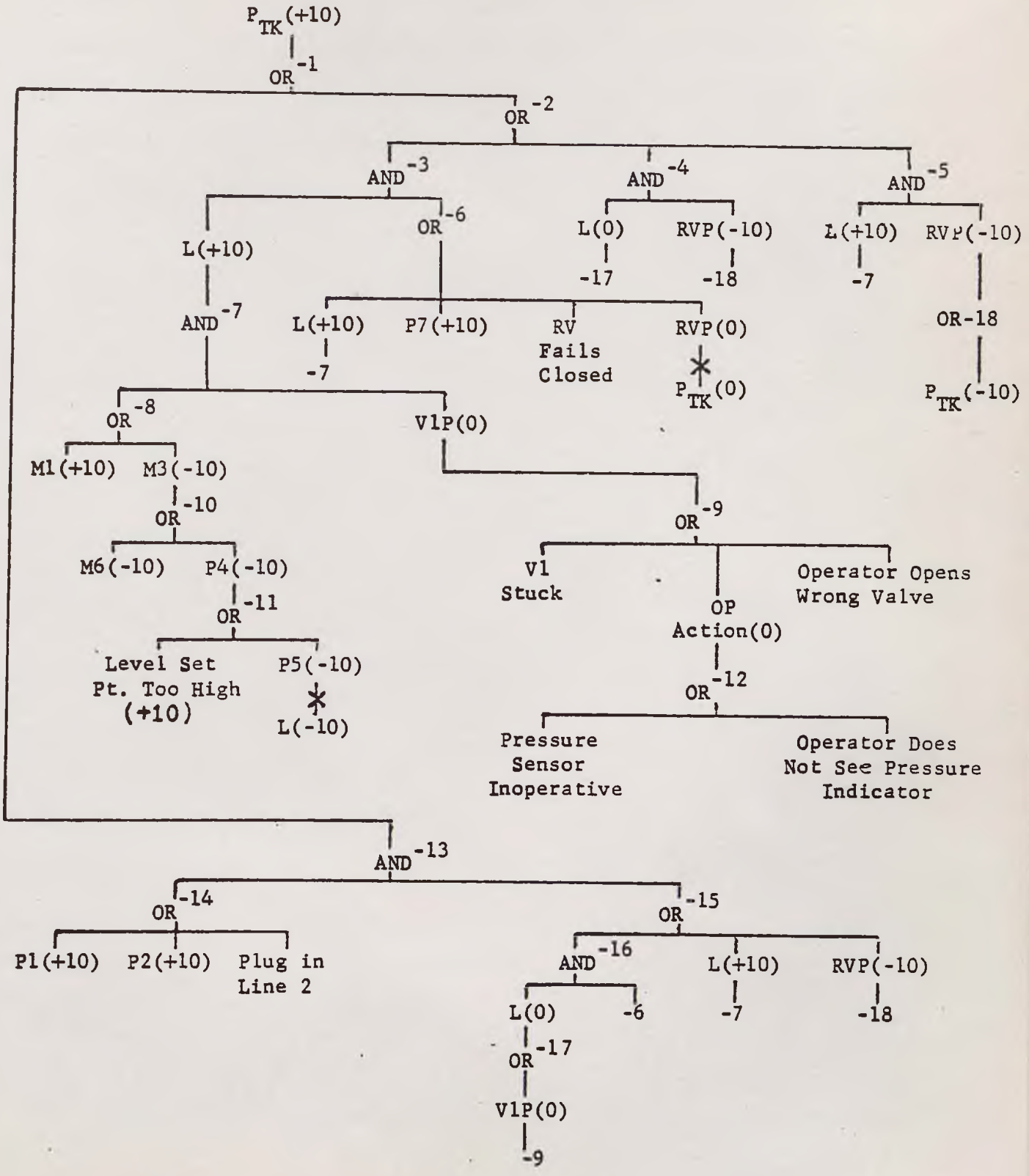


Figure 45

this reason, we do not expect it to cancel the disturbance, and we only analyze it as a possible disturbance source.

With the extensions presented in the last two sections, we can now analyze a large range of systems. Since the preocedure for synthesizing the fault tree is algorithmic, it may be possible to program it so that the digital computer can be used for assistance. This topic is discussed in the next section.

COMPUTER ASSISTED FAULT TREE SYNTHESIS

A computer program called FTS (Fault Tree Synthesis) has been written which constructs fault trees using the algorithm discussed previously. The operations performed by the program are exactly those described in the section describing the algorithm. Before using the computer, however, a way must be developed for conveying all of the information in the digraph to the program. What different pieces of information are contained in the graph? The nodes of the digraph represent process variables and events. For example, nodes such as T4 denote variables and those like PUMP SHUTDOWN represent events. During the course of the synthesis, these nodes will take on values. The value associated with an event node denotes whether or not the event occurs. Nodes having inputs are termed "connecting nodes". Some nodes have no inputs. These correspond to either primal variables or events. The edges of the digraph show how the nodes (variables or events) are connected. Gains associated with the edges determine how one node affects others. In addition, certain types of edges are conditional upon other events (REVERSED VALVE ACTION is an example).

Therefore, classify each node according to two different criteria, the first being whether it corresponds to an event or variable and second whether it is primal or connecting. Edges can be grouped into two classes, those which are conditional on some other event and those which

are not. In order to identify the various types of nodes and edges on the digraph, numbers are assigned to them according to the following table.

Name	Type	Number
Node	Connecting Variable	1ØØ1-N
Node	Connecting Event	1ØØ1-N
Node	Primal Variable	(N+1) and up
Node	Primal Event	(M+1)-1ØØØ
Edge	Conditional	1-M
Edge	Unconditional	(Not Numbered)

Table 2: Digraph Coding Key

Thus one first assigns all conditional edge values of 1 through M. Next any nodes corresponding to primal events are numbered beginning with M+1. Then any nodes denoting connecting variables or events are assigned values of 1ØØ1 through N. Finally any remaining nodes corresponding to primal variables are numbered beginning with N+1.

Input to FTS then takes the following form. First the values of M and (N-1ØØØ) are listed. These provide information on how many conditional edges and connecting nodes exist. Next, the inputs to each connecting node are

listed along with their appropriate gains. If input is via an unconditional edge, the value of the node from which the edge originates is listed. If it is by a conditional edge, the value of the edge itself is used. The gain associated with each edge is listed immediately below its value. This block of data allows FTS to determine the connectivity within the graph. Next the nodes from which each conditional edge originates are listed. Together with the first block of data, this allows the program to determine special cases of "conditional connectivity". Finally the node corresponding to the top event is listed along with its value. This tells FTS where to begin on the digraph.

Figure 46 shows the digraph for the HNO3 problem numbered for input to FTS. In figure 47 , the actual input to the program is listed. Run time for this problem was approximately one second on an IBM 360/67 computer. The output from FTS is shown in figure 48 and a tree constructed from it follows in figure 49. Note the feedback structure in gate 1 and the feedforward construction in gate 13.

FTS is a prototype system and has not been tested extensively. A copy of it is on file in the Department of Chemical Engineering at Carnegie-Mellon University. Appendix B contains an article [12] describing a larger system on which it was used.

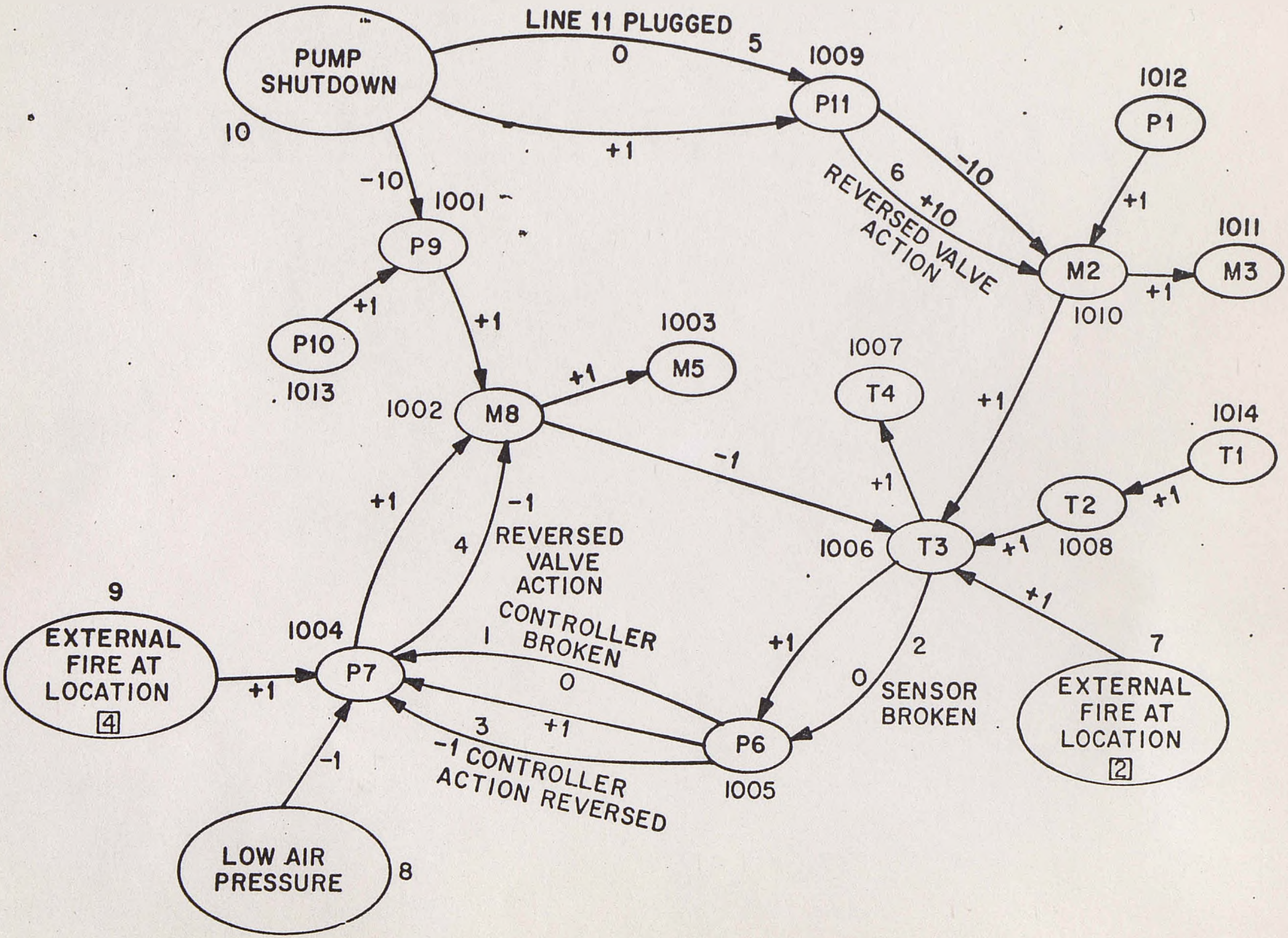


Figure 46: Numbered Digraph for HNO3 Process

```
11  6        - Number of connecting nodes and conditional edges
2            - Number of inputs to node 1001
  10 1013    - Inputs to node 1001
 -10    1    - Gains associated with inputs to node 1001
3
1001 1004     4
   1    1    -1
1
1002
   1
5
   9    8     3 1005    1
   1   -1    -1    1    0
2
   2 1006
   0    1
4
1002 1010 1008    7
  -1    1    1    1                    Connectivity Information
1                                       for Nodes 1002 - 1011
1006
   1
1
1014
   1
2
   5   10
   0    1
3
   6 1009 1012
  10  -10    1
1
1010
   1
1005   - Origin of Conditional Edge 1
1006
1005                                    Origin of Conditional Edges 2-6
1004
  10
1009
1006    1  - Definition of Top Event
```

Figure 47: Input For FTS Program

```
GATE NUMBER    1                          GATE NUMBER    2
 1006    OR      -6   -5   -2              0    AND      -3   -4
    1             0    0    0              0              0    0

GATE NUMBER    3                          GATE NUMBER    4
    0    OR    1012 1014    7              0    OR      -12   -5
    0           1    1     1              0              0    0

GATE NUMBER    5                          GATE NUMBER    6
 1002    OR    -13 1013   -7   -8          0    OR    1012 1014    7
   -1           0  -10    0    0          0           10   10   10

GATE NUMBER    7                          GATE NUMBER    8
    0   EOR     -10    4                   0    AND      -16   -9
    0             0    1                   0              0    0

GATE NUMBER    9                          GATE NUMBER   10
    0    OR      -12   -7                 1004    OR      -11    8    3
    0             0    0                   -1             0   10    1

GATE NUMBER   11                          GATE NUMBER   12
    0   AND      -15    8                   0    OR        1    2
    0             0    1                   0              1    1

GATE NUMBER   13                          GATE NUMBER   14
    0   AND      -14   10                   0    OR        5    6
    0             0    1                   0              1    1

GATE NUMBER   15                          GATE NUMBER   16
    0    OR      -12    3                   0    OR    1013  -13
    0             0    1                   0             -1    0
```

Figure 48 : Output From FTS For HNO3 Problem
(Negative numbers indicate gate connections. Numbers under gate inputs are values associate with primal events. Gate connections have no associated values.)

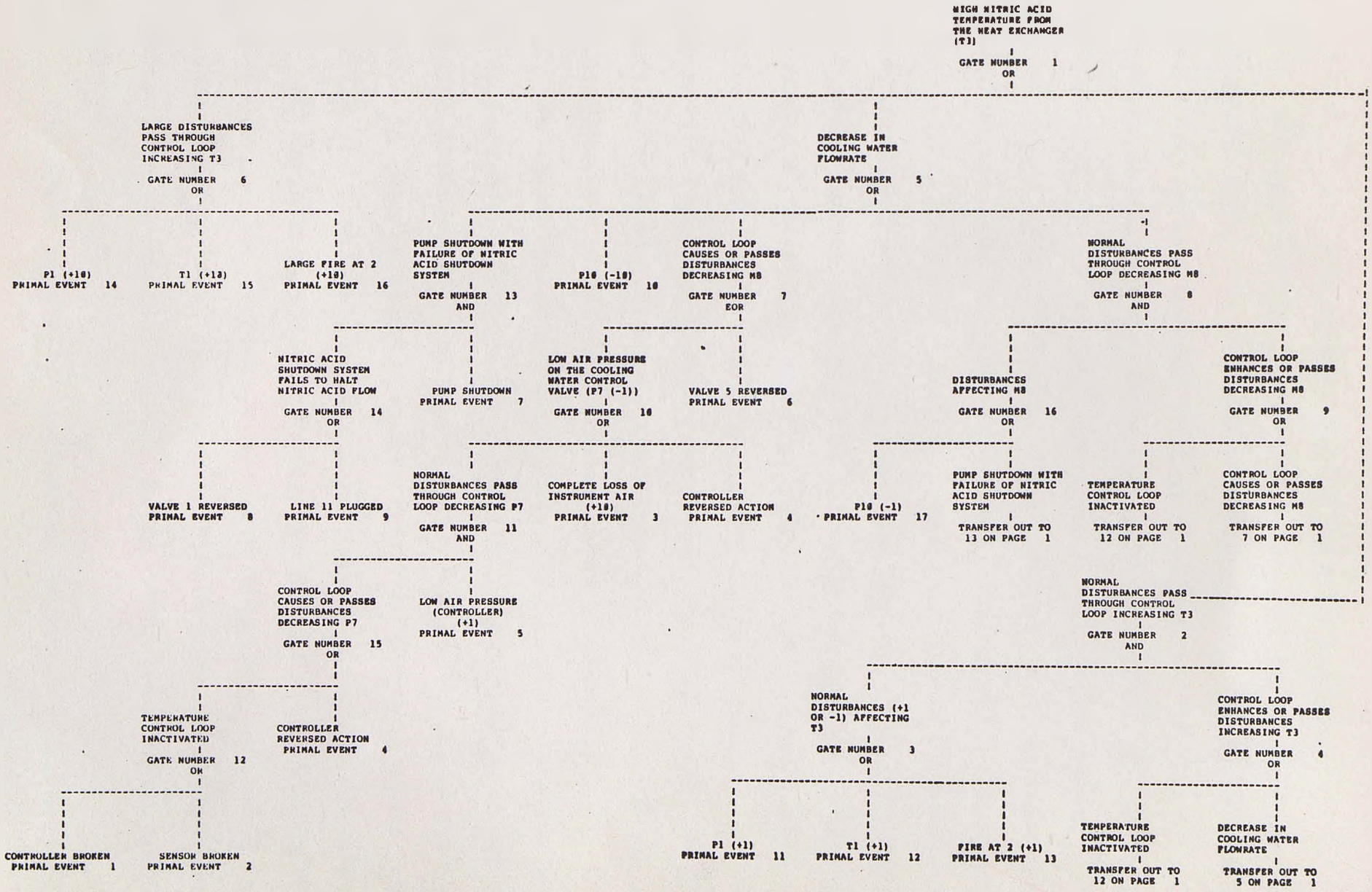


Figure 49: Fault Tree Constructed from FTS Output

CONCLUSIONS AND RECOMMENDATIONS

A formal method of modelling process components for use in fault tree analysis has been developed. Digraph models are able to capture the behavior of components as well as the procedures used by operators in running a process. Sequential behavior can be captured easily using the conditional edge feature. The models developed in this thesis are for chemical process components. Digraphs, however, can also be used to capture cause and effect in electrical, mechanical, and nuclear systems. Since this is the case, these systems can also be analyzed using the procedures developed in this thesis. Different levels of modelling are also possible. Hence, simple analyses may be done with simple models and more complex models reserved for more detailed studies. Digraph models also allow a convenient way of storing expertise and experience.

An algorithmic procedure for fault tree synthesis has been defined. The procedure focuses the analyst's thinking on the important interactions in the system, these being the loops in the digraph. Fault trees synthesized using the method are highly structured. This is quite important since it allows the analyst to easily modify the trees to reflect changes in the system. It also suggests a basis of standardization for fault trees. With practice, the time needed to synthesize fault trees can be significantly reduced.

In order that this method be useful, it is critical

that high quality models be available to the analyst. Since the algorithm is an information processor, fault trees developed using it will only be as good as the models used to describe the system. It is recommended that future work be aimed at the development of these models. Data for the equipment models could be gathered from personnel who have expertise concerning various pieces of equipment. A high quality library of models could be built in this way and would be of considerable value in itself. Operating procedures will vary from system to system so it would be useful to have a language whereby the procedures could be easily translated into a digraph model. This model could then be linked along with those from the equipment library to from the system digraph. Having done this, the analyst could proceed with the construction of the fault tree using a graph that has the expertise of a number of different people stored in it. With this, high quality fault trees can be developed without expending a large amount of time.

BIBLIOGRAPHY

[1] Haasl, D.F., "Advanced Concepts in Fault Tree Analysis", System Safety Symposium, June 8-9, 1965, Seattle: The Boeing Company.

[2] Vesely, W.E. and Narum, R.E., "PREP and KITT: Computer Codes for the Automatic Evaluation of a Fault Tree", IN-1349, August (1970).

[3] Fussell, J.B. and Vesely, W.E., "A New Methodology for Obtaining Cut Sets for Fault Trees", Transactions of the American Nuclear Society, 15, (1972).

[4] Reactor Safety Study - An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants, WASH-1400 (NUREG-75/014), U.S. Nuclear Regulatory Commission, Washington, D.C., October, 1975.

[5] Powers, G.J., "Advanced Process Synthesis" - Course Notes, Carnegie-Mellon University, 1974.

[6] Fussell, J.B., "Synthetic Tree Model - A Formal Methodology for Fault Tree Construction", Aerojet Nuclear Report ANCR 01098, March (1973).

[7] Taylor, J.R., "A Formalisation of Failure Mode Analysis of Control Systems", Danish Atomic Energy Commission, RISO-M-1654, September (1973).

[8] Tompkins, F.C. and Powers, G.J., "Fault Tree Synthesis for Chemical Processes", AICHE Journal, 20(2), March (1974).

[9] Salem, S.L., "A Computer Oriented Approach to Fault Tree Construction", UCLA-ENG-7635, April (1976).

[10] Kaplan, W., Advanced Calculus, Addison-Wesley Publishing Co., Reading, Mass. (1959), pp. 90-97.

[11] Schaewitz, J.A., Lapp, S.A., and Powers, G.J., "Fault Tree Analysis of Sequential Systems", I&EC Process Design and Development, 16(4), (1977).

[12] Powers, G.J. and Lapp, S.A., "Computer Aided Fault Tree Synthesis", Chemical Engineering Progress 72(4), April (1976).

APPENDIX A

PLEASE NOTE:

Pages 98-124 contain very small print. Filmed as received in the best possible way.

UNIVERSITY MICROFILMS INTERNATIONAL

Joseph A. Shaelwitz, Steven A. Lapp, and Gary J. Powers*

Department of Chemical Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213

Fault tree analysis is a systems safety technique for determining the logical combinations of events which could cause a specific hazard to occur. Digraph models are proposed which describe sequential relationships between events. A computer program which automatically constructs fault trees from digraph models is illustrated for an air-drying process.

Introduction

Fault tree analysis has been used in the aerospace, electronics, nuclear, and chemical industries to aid in (1) the discovery and control of failures before they occur; (2) the analysis of accidents, and (3) the planning of maintenance activities (Fussell, 1974; Powers, 1974). In the use of the fault tree method many safety analysts have expressed concern over the ability of the method to handle sequentially dependent events.

Esary recently illustrated the problem of applying fault tree analysis to sequential systems (Esary and Ziehms, 1975). In his analysis he considered a phased-mission in which the status and function of the components within a system depend on the phase (time sequence) of the mission. Esary presented a fault tree for the phased mission which was composed of sub-trees, one for each phase in the mission. He also presented an excellent discussion of the difficulties involved in calculating the probability of system success given the sequential interdependence of events. In this paper we present a strategy for partially automating the synthesis of fault trees when sequential interdependencies are considered. The key to the method is the development of cause and effect models which explicitly consider the possible sequential interactions between events. These models are used in an algorithm which automatically generates fault trees.

Digraph Models

In a previous paper (Lapp, 1977), we have developed the concept of a digraph model for the description of both the normal and failed behavior of components in interconnected systems. The models will be briefly reviewed here and extended to sequential situations.

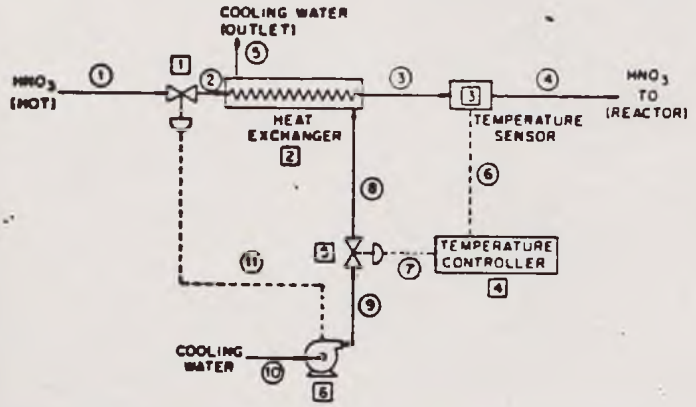


Figure 1. Flow diagram for part of a nitrification process. When low pump speed is sensed at the pump, valve 1 is closed.

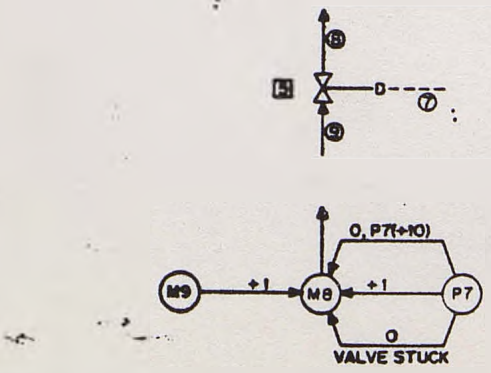


Figure 2. A cause-and-effect model for a control valve.

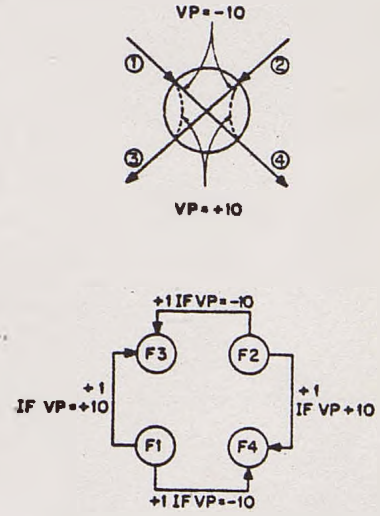


Figure 3. Cause-and-effect model for a four-way valve.

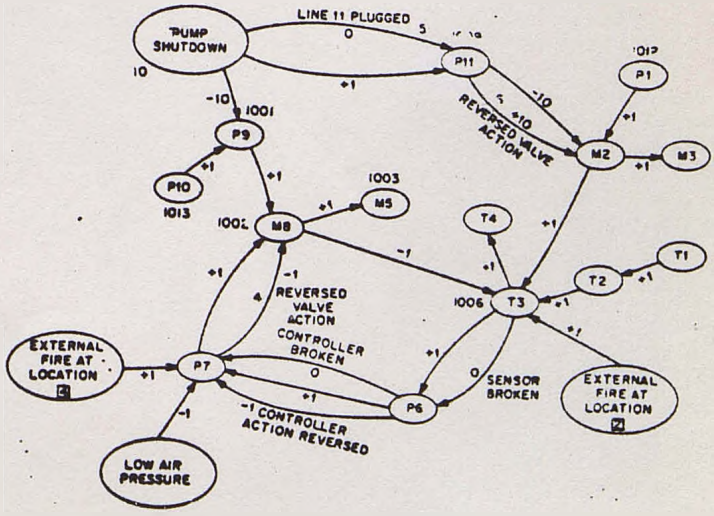


Figure 4. A partial cause-and-effect model for the output variable temperature in stream 4 (T4) in Figure 1.

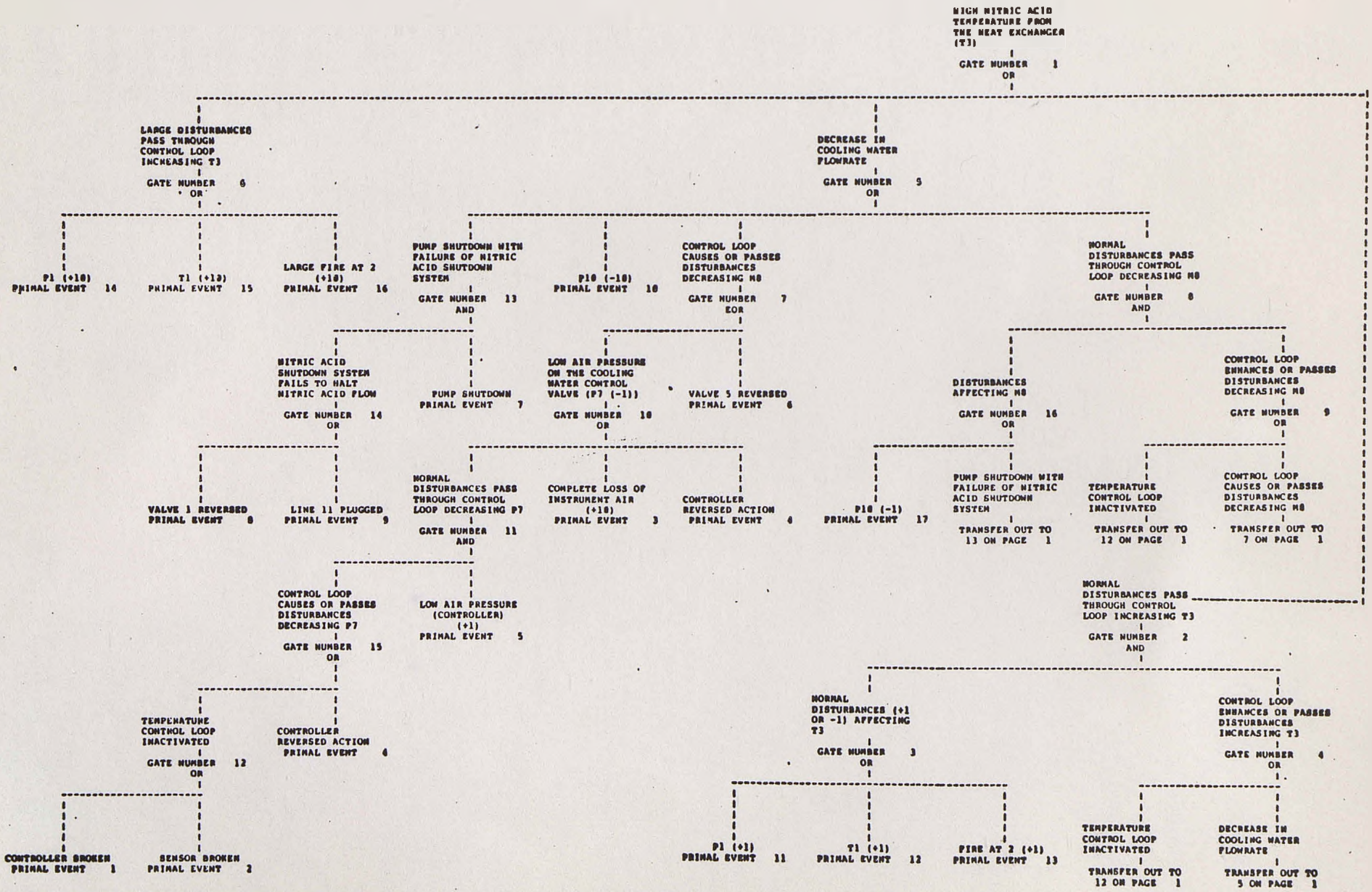


Figure 5. Fault tree for the event T4 too high.

Table I. Operating Procedure for Fixed-Bed Drier System

Time period	Time since beginning of cycle, h	Value position (steam connections) 3W	4WI	4WII	Bed status Bed I	Bed II
1	1, 2	11→12	18→19 AND 22→23	20→21 AND 24→25	Regeneration	In service
2	3	11→18	18→19 AND 22→23	20→21 AND 24→25	Cooling	In service
3	4, 5	11→12	18→23 AND 22→19	20→25 AND 24→21	In service	Regeneration
4	6	11→18	18→23 AND 22→19	20→25 AND 24→21	In service	Cooling

Return to Time Period 1

In developing logic models for physical systems, it is necessary to capture the cause-and-effect nature of interactions which occur between the variables which describe the system (i.e., temperatures, pressures, flow rates, concentrations, operator action, voltages, valve positions, etc.) and events which occur within the system (i.e., valve failure, fire, explosion, operator error, weather changes, etc.). For example, consider the flow sheet given in Figure 1. The system is composed of valves, pumps, heat exchangers, etc. In order to determine how failures or deviations in input variables propagate through the system, it is possible to construct digraphs whose nodes represent events or variables and whose edges represent the relationships between the nodes. Figure 2 illustrates a cause and effect model for valve 5 in this system. Note that the edges may be event dependent. That is, a relationship between two variables may be dependent on the value of other variables or events in the system. For example, the gain between the pressure on the valve actuator P7 and the mass flow rate leaving the valve M8 is normally +1. An increase in pressure opens the valve and allows a higher flow rate. However, if the pressure on the actuator is very high (P7 = +10) the gain is 0. The valve is wide open and further increases in pressure P7 do not give an increase in the flow rate M8.

If the edges in a digraph are made dependent on other events in the system, it is possible to capture in one digraph the sequential behavior of the complete system. Consider the four-way valve shown in Figure 3. When the valve is in position 1 (valve position = VP = +10) the flow is from stream 1 to stream 3 and from stream 2 to stream 4. In position 2 (valve position = VP = −10) the flow is from stream 2 to stream 3 and from stream 1 to stream 4. A digraph for this valve is given in Figure 3 where VP is the valve position. If, from another

Table II. Sequential Dryer Process Events

Title event number	Probability	Text
1	2.00×10^{-3}	4-Way valve leaks across
2	5.00×10^{-8}	Fire at Bed I
3	1.20×10^{-4}	No alumina in Bed I, or channeling
4	7.15×10^{-4}	4-Way valve II motor failure
5	3.30×10^{-1}	Time = 1
6	1.67×10^{-1}	Time = 2
7	4.72×10^{-4}	4-Way valve II timer changes at wrong time
8	5.00×10^{-8}	Fire at Bed II
9	1.20×10^{-4}	No alumina in Bed II, or channeling
10	1.67×10^{-1}	Time = 4
11	4.72×10^{-4}	4-Way valve II timer fails
12	8.63×10^{-5}	4-Way valve II control line 29 cut
13	3.30×10^{-1}	Time = 3
14	5.00×10^{-4}	Inlet air flow up
15	5.00×10^{-4}	Proportionating valve pressure up (P10)
16	7.15×10^{-4}	4-Way valve I motor failure
17	7.99×10^{-5}	Heater leaks steam into air
18	5.00×10^{-4}	Inlet air water concentration up
19	7.15×10^{-4}	3-Way valve motor-failure
20	4.72×10^{-4}	4-Way valve I timer changes at wrong time
21	4.72×10^{-4}	4-Way valve I timer fails
22	8.63×10^{-5}	4-Way valve I control line 28 cut
23	1.00×10^{-5}	Water separator trap clogged
24	2.10×10^{-8}	Valve 6 closed
25	5.00×10^{-8}	External fire at separator
26	4.72×10^{-4}	3-Way valve timer changes at wrong time
27	4.72×10^{-4}	3-Way valve timer fails
28	8.63×10^{-5}	3-Way valve control line 27 cut
29	5.00×10^{-4}	Cooling water flow down
30	5.00×10^{-4}	Cooling water temperature up
31	2.99×10^{-3}	Cooler fouled
32	5.00×10^{-8}	External fire at cooler
33	2.10×10^{-3}	Valve 4 closed
36	5.00×10^{-4}	Inlet Air pressure up

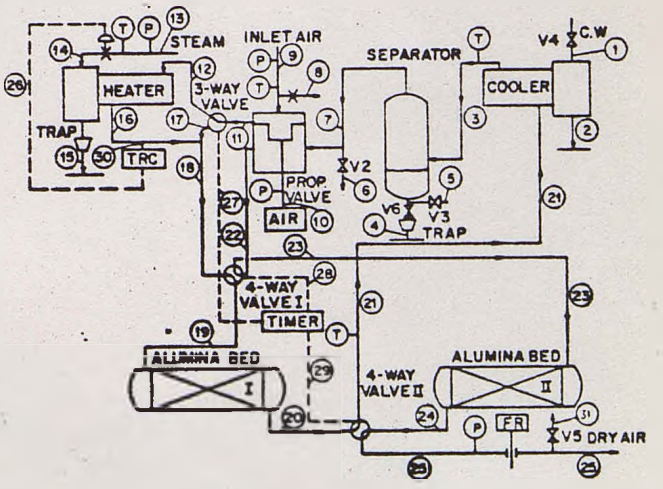
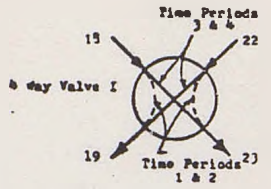


Figure 6. Flow diagram of a utility air drying process. After King (1970).

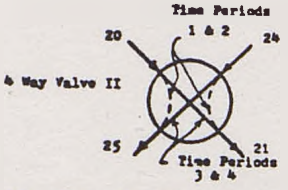


	F19	T19	C19	F23	T23	C23
F18	+1					
T18		+1				
C18			+1			
F22				+1		
T22					+1	
C22						+1
Failures: Plug	-10			-10		
Leak Across	- 1			+ 1		
Leak Across	C18 to C13 Gain +1, T18 to T23 Gain +1					

Time Periods 1 AND 2
VP = +10

	F19	T19	C19	F23	T23	C23
F18				+1		
T18					+1	
C18						+1
F22	+1					
T22		+1				
C22			+1			
Failures: Plug	-10			-10		
Leak Across	+ 1			- 1		
Leak Across	C18 to C19 Gain +1, T18 to T19 Gain +1					

Time Periods 3 AND 4
VP = -10



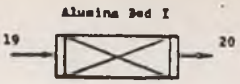
	F21	T21	C21	F25	T25	C25
F20	+1					
T20		+1				
C20			+1			
F24				+1		
T24					+1	
C24						+1
Plug	-10			-10		
Leak Across	- 1			+ 1		
Leak Across	C20 to C25 Gain +1, T20 to T25 Gain +1					

Time Periods 1 AND 2
VP = +10

	F21	T21	C21	F25	T25	C25
F20				+1		
T20					+1	
C20						+1
F24	+1					
T24		+1				
C24			+1			
Plug	-10			-10		
Leak Across	+ 1			- 1		
Leak Across	C20 to C21 Gain +1, T20 to T21 Gain +1					

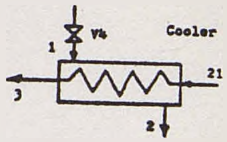
Time Periods 3 AND 4
VP = -10

	VP = +10				VP = -10			
	F12	T12	C12	F17	F17	T17	C17	F12
F11	+1				+1			
T11		+1				+1		
C11			+1				+1	
Failures: Valve Leaks Across	-1			+1	-1			+1
	Use VP = -10 Gains for T and C				Use VP = +10 Gains for T and C			
Plugged Valve	-10				-10			

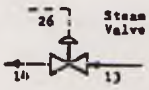


	Time Period 1 (Regeneration)			Time Period 2 (Cooling)			Time Period 3 AND 4 (In Service)		
	F20	T20	C20	F20	T20	C20	F20	T20	C20
F19	+1	-1	-1	+1	+1	0	+1	0	+1
T19	0	+1	+1	0	+1	0	0	+1	+1
C19	0	0	+1	0	0	+1	0	0	+1
Failures									
No alumina in Bed I (plus channeling)	0	+1	-1	0	+1	0	0	0	+1
Plugged Bed	-10	0	0	-10	0	0	-10	0	0
Leak Out	- 1	0	0	- 1	0	0	- 1	0	-1
Fire	0	+1	+1	0	+1	0	0	+1	+1

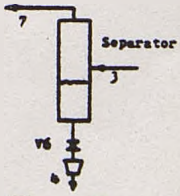
	Time Period 1 AND 2 (In Service)			Time Period 3 (Regeneration)			Time Period 4 (Cooling)		
	F24	T24	C24	F24	T24	C24	F24	T24	C24
F23	+1	0	+1	+1	-1	-1	+1	+1	0
T23	0	+1	+1	0	+1	+1	0	+1	0
C23	0	0	+1	0	0	+1	0	0	+1
Failures	0	0	+1	0	+1	-1	0	+1	0
No alumina in Bed II (channeling)	0	0	+1	0	+1	-1	0	+1	0
Plugged Bed	-10	0	0	-10	0	0	-10	0	0
Leak Out	- 1	0	-1	- 1	0	0	- 1	0	0
Fire	0	+1	+1	0	+1	+1	0	+1	0



	F2	T2	C2	F1	T1	C1
F1	+1	-1	0	-1	-1	
T1	0	+1	0	+1	+1	
F21	0	+1	+1	+1	+1	
T21	0	+1	0	+1	+1	
C21	0	0	0	0	+1	
Failures						
External Fire	0	+1	0	+1	+1	
Fouling	0	+1	0	+1	+1	
V4 Open Wide	+10	-10	0	-10	-10	
V4 Closed	-10	+10	0	+10	+10	



	F14	T14
F26	-1	0
F13	+1	0
T13	0	+1
Failures: Line 13 plugged	-10	0
Valve Reversed	Change F26 to F14 Gain to +1	
Valve Fails Closed	-10	0
Valve Fails Open	+10	0



	F7	T7	C7	F4	T4
F3	+1	0	0	+1	0
T3	0	+1	+1	0	+1
C3	0	0	+1	0	0
Failures					0
Stem Plugged	+1	0	+1	-1	-1
Fire	0	+1	+1	-10	+1
V6 Closed	+1	0	+1	-10	0

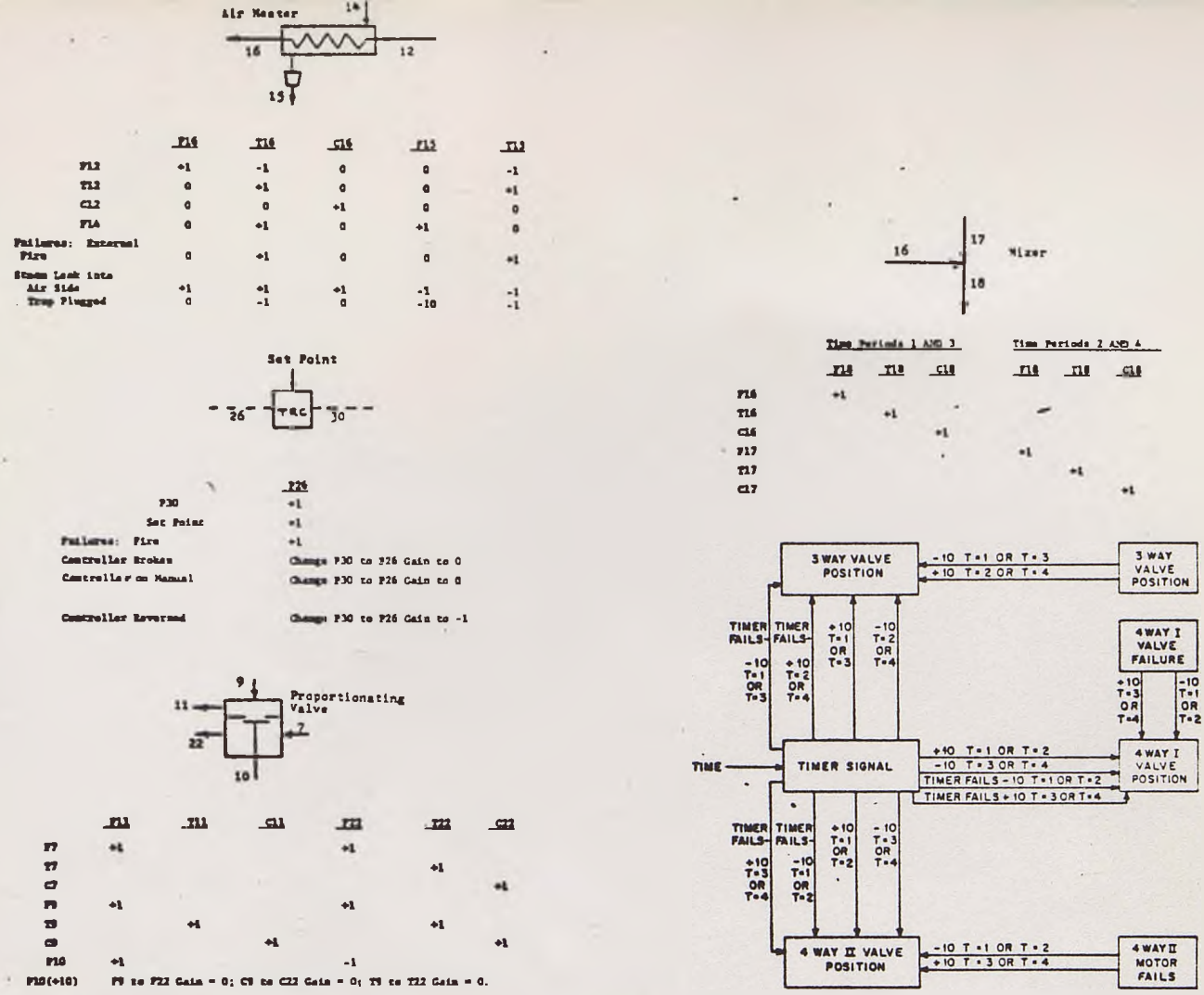


Figure 7. Input–output models for the equipment in the utility air drying process.

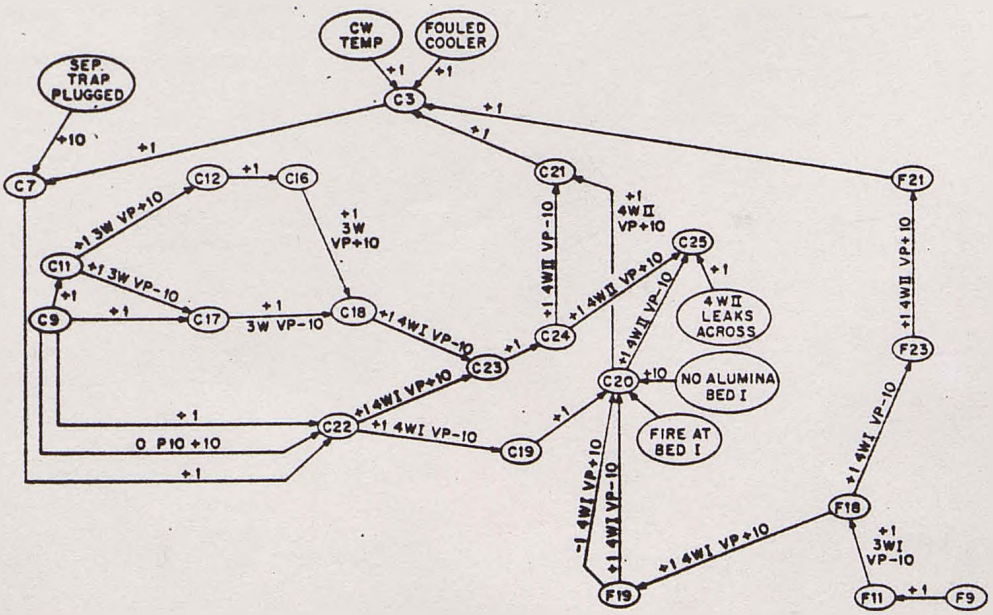
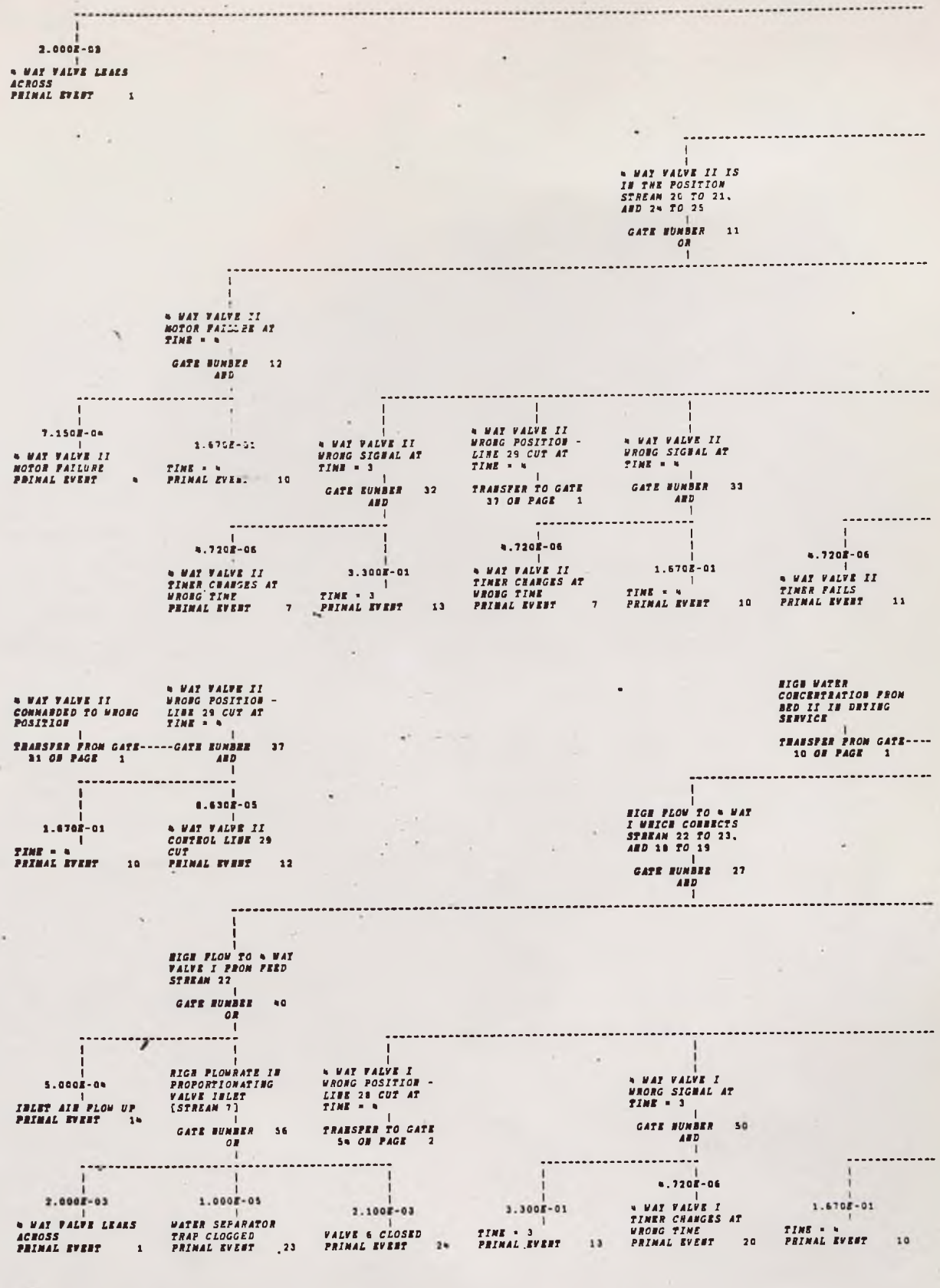


Figure 8. Part of the digraph for the utility air drying process.

part of the system, the position of the four-way valve is commanded to a particular sequence of positions, these sequences will be conditionals which modify the relationships between F1, F2, F3, and F4. For example, the event F3 (+1) equals (FI (+1) AND VP (+10)) OR (F2 (+1) AND VP (−10)). If VP (+10) is dependent on other events they may be expanded into the appropriate Boolean expression and substituted for VP (+10).

Digraph models of this type are able to describe both combinational and sequential logic relationships. They are much more compact than truth tables, decision tables, or finite state models. In addition, the Boolean logic is computed from


2.000E-03
4 WAY VALVE LEAKS ACROSS PRIMAL EVENT 1
4 WAY VALVE II IS IN THE POSITION STREAM 20 TO 21, AND 24 TO 25
GATE NUMBER 11
OR
4 WAY VALVE II MOTOR FAILURE AT TIME = 4
GATE NUMBER 12
AND
7.150E-04
4 WAY VALVE II MOTOR FAILURE PRIMAL EVENT 4
1.670E-01
TIME = 4 PRIMAL EVENT 10
4 WAY VALVE II WRONG SIGNAL AT TIME = 3
GATE NUMBER 32
AND
4 WAY VALVE II WRONG POSITION - LINE 29 CUT AT TIME = 4
TRANSFER TO GATE 37 ON PAGE 1
4 WAY VALVE II WRONG SIGNAL AT TIME = 4
GATE NUMBER 33
AND
4.720E-06
4 WAY VALVE II TIMER CHANGES AT WRONG TIME PRIMAL EVENT 7
3.300E-01
TIME = 3 PRIMAL EVENT 13
4.720E-06
4 WAY VALVE II TIMER CHANGES AT WRONG TIME PRIMAL EVENT 7
1.670E-01
TIME = 4 PRIMAL EVENT 10
4.720E-06
4 WAY VALVE II TIMER FAILS PRIMAL EVENT 11
4 WAY VALVE II CONNECTED TO WRONG POSITION
TRANSFER FROM GATE 31 ON PAGE 1
4 WAY VALVE II WRONG POSITION - LINE 29 CUT AT TIME = 4
GATE NUMBER 37
AND
1.670E-01
TIME = 4 PRIMAL EVENT 10
8.630E-05
4 WAY VALVE II CONTROL LINE 29 CUT PRIMAL EVENT 12
HIGH WATER CONCENTRATION FROM BED II IN DRYING SERVICE
TRANSFER FROM GATE 10 ON PAGE 1
HIGH FLOW TO 4 WAY I WHICH CORRECTS STREAM 22 TO 23, AND 18 TO 19
GATE NUMBER 27
AND
HIGH FLOW TO 4 WAY VALVE I FROM FEED STREAM 22
GATE NUMBER 40
OR
5.000E-04
INLET AIR FLOW UP PRIMAL EVENT 14
RICH FLOWRATE IN PROPORTIONATING VALVE INLET (STREAM 7)
GATE NUMBER 56
OR
4 WAY VALVE I WRONG POSITION - LINE 28 CUT AT TIME = 4
TRANSFER TO GATE 54 ON PAGE 2
4 WAY VALVE I WRONG SIGNAL AT TIME = 3
GATE NUMBER 50
AND
2.000E-03
4 WAY VALVE LEAKS ACROSS PRIMAL EVENT 1
1.000E-05
WATER SEPARATOR TRAP CLOGGED PRIMAL EVENT 23
2.100E-03
VALVE 6 CLOSED PRIMAL EVENT 24
3.300E-01
TIME = 3 PRIMAL EVENT 13
4.720E-06
4 WAY VALVE I TIMER CHANGES AT WRONG TIME PRIMAL EVENT 20
1.670E-01
TIME = 4 PRIMAL EVENT 10
PAGE 11
SEQUENTIAL DRYER PROCESS

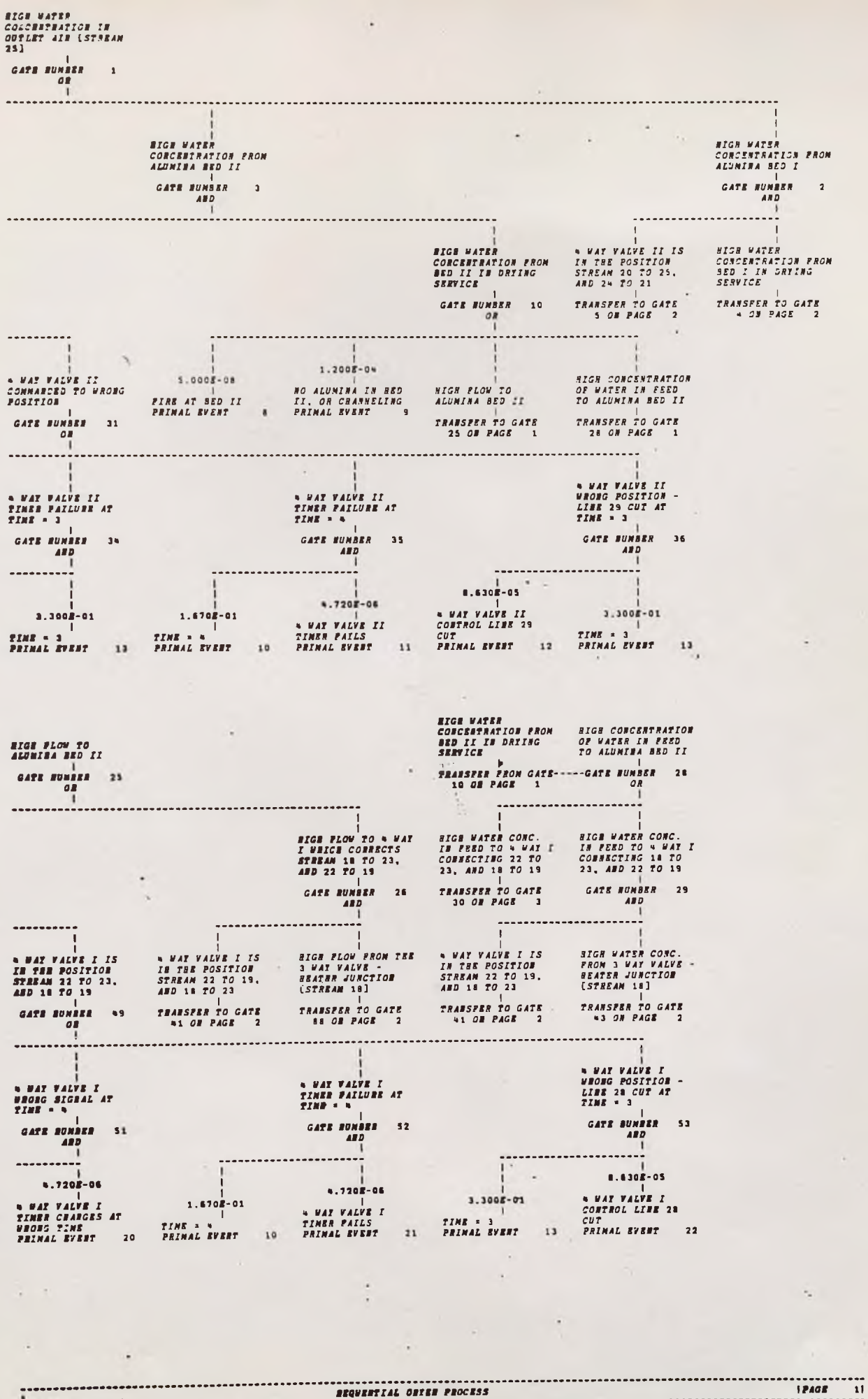
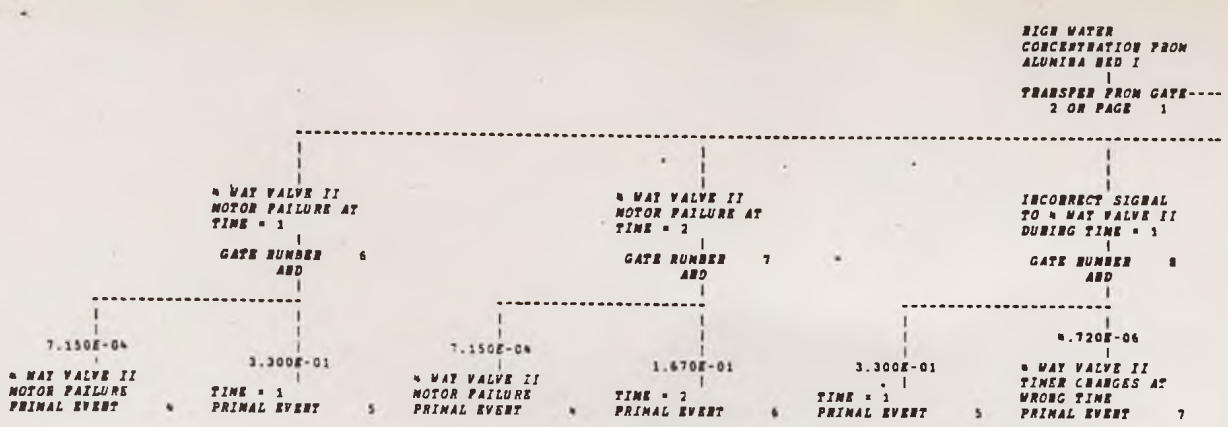
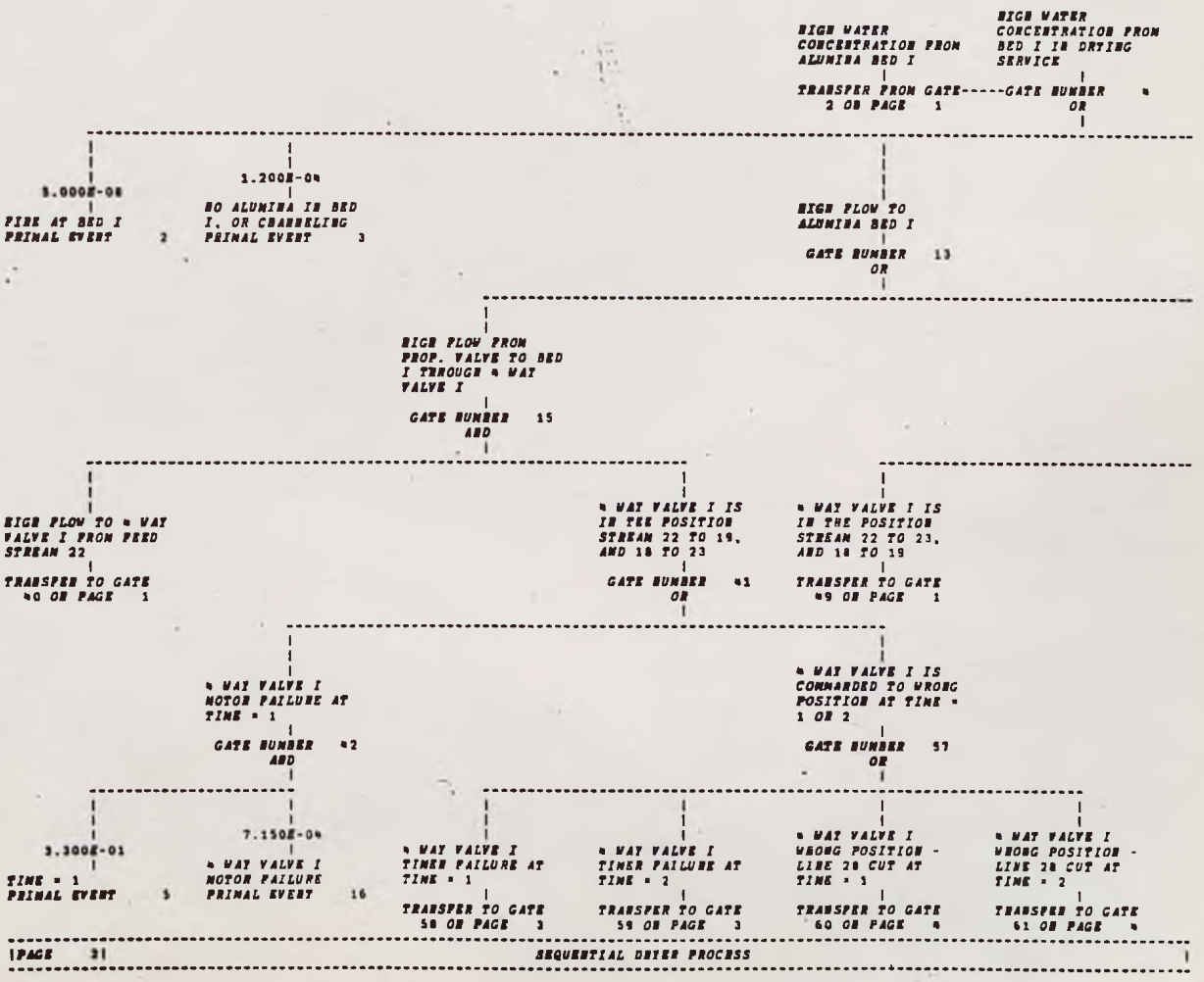
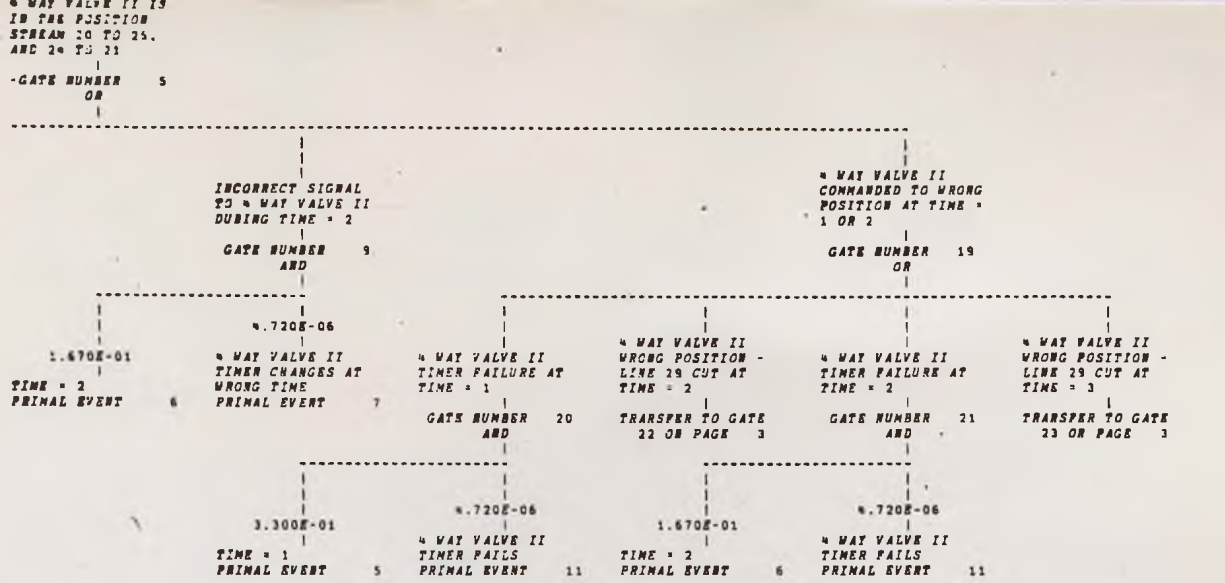


Figure 9 *continues*


HIGH WATER CONCENTRATION FROM ALUMINA BED I
TRANSFER FROM GATE 2 ON PAGE 1
4 WAY VALVE II MOTOR FAILURE AT TIME = 1
GATE NUMBER 6 AND
4 WAY VALVE II MOTOR FAILURE AT TIME = 2
GATE NUMBER 7 AND
INCORRECT SIGNAL TO 4 WAY VALVE II DURING TIME = 1
GATE NUMBER 8 AND
7.150E-04
4 WAY VALVE II MOTOR FAILURE PRIMAL EVENT 4
3.300E-01
TIME = 1 PRIMAL EVENT 5
7.150E-04
4 WAY VALVE II MOTOR FAILURE PRIMAL EVENT 4
1.670E-01
TIME = 2 PRIMAL EVENT 6
3.300E-01
TIME = 1 PRIMAL EVENT 5
4.720E-06
4 WAY VALVE II TIMER CHANGES AT WRONG TIME PRIMAL EVENT 7


HIGH WATER CONCENTRATION FROM ALUMINA BED I
TRANSFER FROM GATE 2 ON PAGE 1
HIGH WATER CONCENTRATION FROM BED I IN DRYING SERVICE
GATE NUMBER 4 OR
5.000E-08
FIRE AT BED I PRIMAL EVENT 2
1.200E-04
NO ALUMINA IN BED I, OR CHANNELING PRIMAL EVENT 3
HIGH FLOW TO ALUMINA BED I
GATE NUMBER 13 OR
HIGH FLOW FROM PROP. VALVE TO BED I THROUGH 4 WAY VALVE I
GATE NUMBER 15 AND
HIGH FLOW TO 4 WAY VALVE I FROM FEED STREAM 22
TRANSFER TO GATE 40 ON PAGE 1
4 WAY VALVE I IS IN THE POSITION STREAM 22 TO 19, AND 18 TO 23
GATE NUMBER 41 OR
4 WAY VALVE I IS IN THE POSITION STREAM 22 TO 23, AND 18 TO 19
TRANSFER TO GATE 49 ON PAGE 1
4 WAY VALVE I MOTOR FAILURE AT TIME = 1
GATE NUMBER 42 AND
4 WAY VALVE I IS COMMANDED TO WRONG POSITION AT TIME = 1 OR 2
GATE NUMBER 57 OR
3.300E-01
TIME = 1 PRIMAL EVENT 5
7.150E-04
4 WAY VALVE I MOTOR FAILURE PRIMAL EVENT 16
4 WAY VALVE I TIMER FAILURE AT TIME = 1
TRANSFER TO GATE 58 ON PAGE 3
4 WAY VALVE I TIMER FAILURE AT TIME = 2
TRANSFER TO GATE 59 ON PAGE 3
4 WAY VALVE I WRONG POSITION - LINE 20 CUT AT TIME = 1
TRANSFER TO GATE 60 ON PAGE 4
4 WAY VALVE I WRONG POSITION - LINE 20 CUT AT TIME = 2
TRANSFER TO GATE 61 ON PAGE 4
PAGE 21
SEQUENTIAL DRYER PROCESS



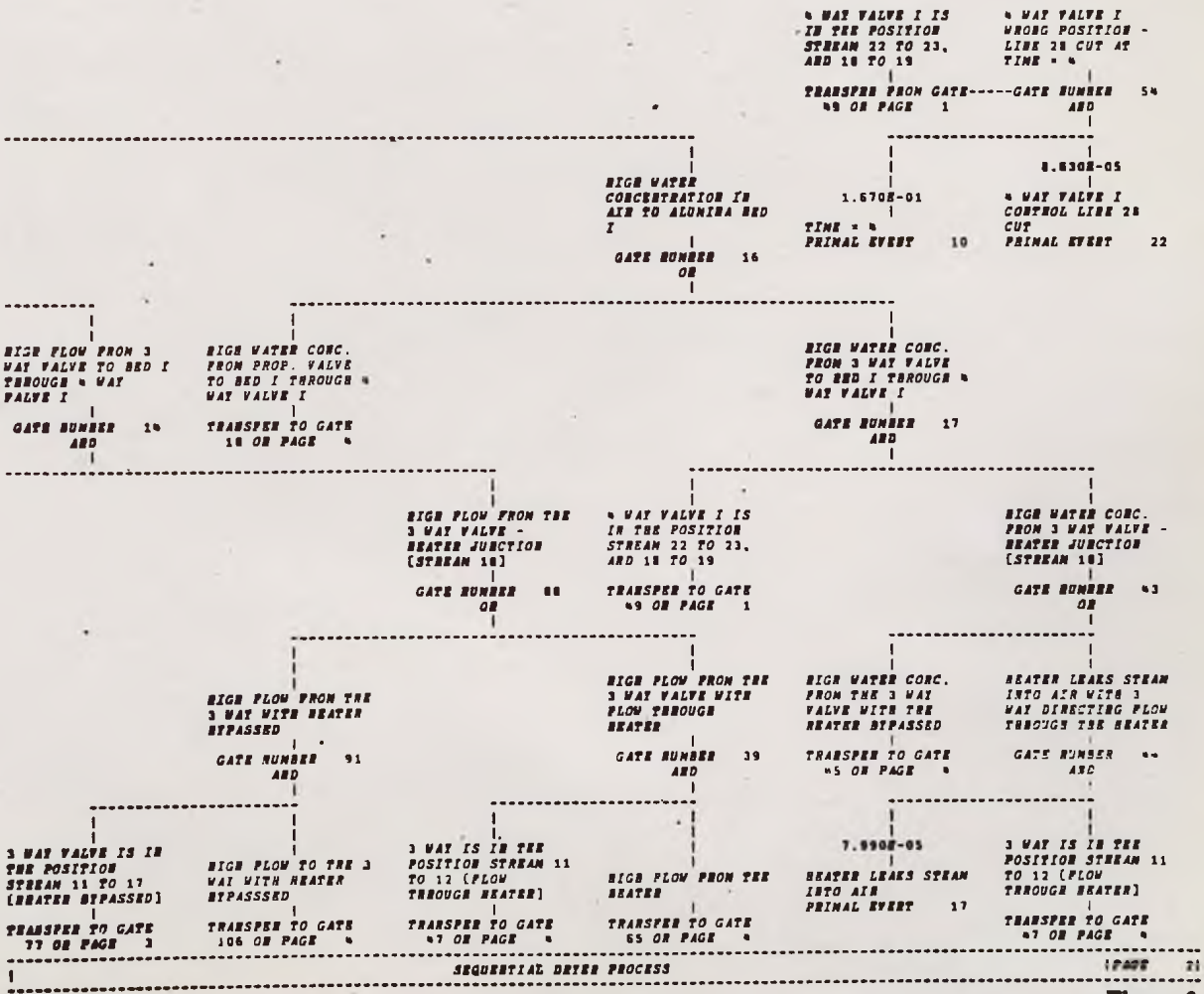
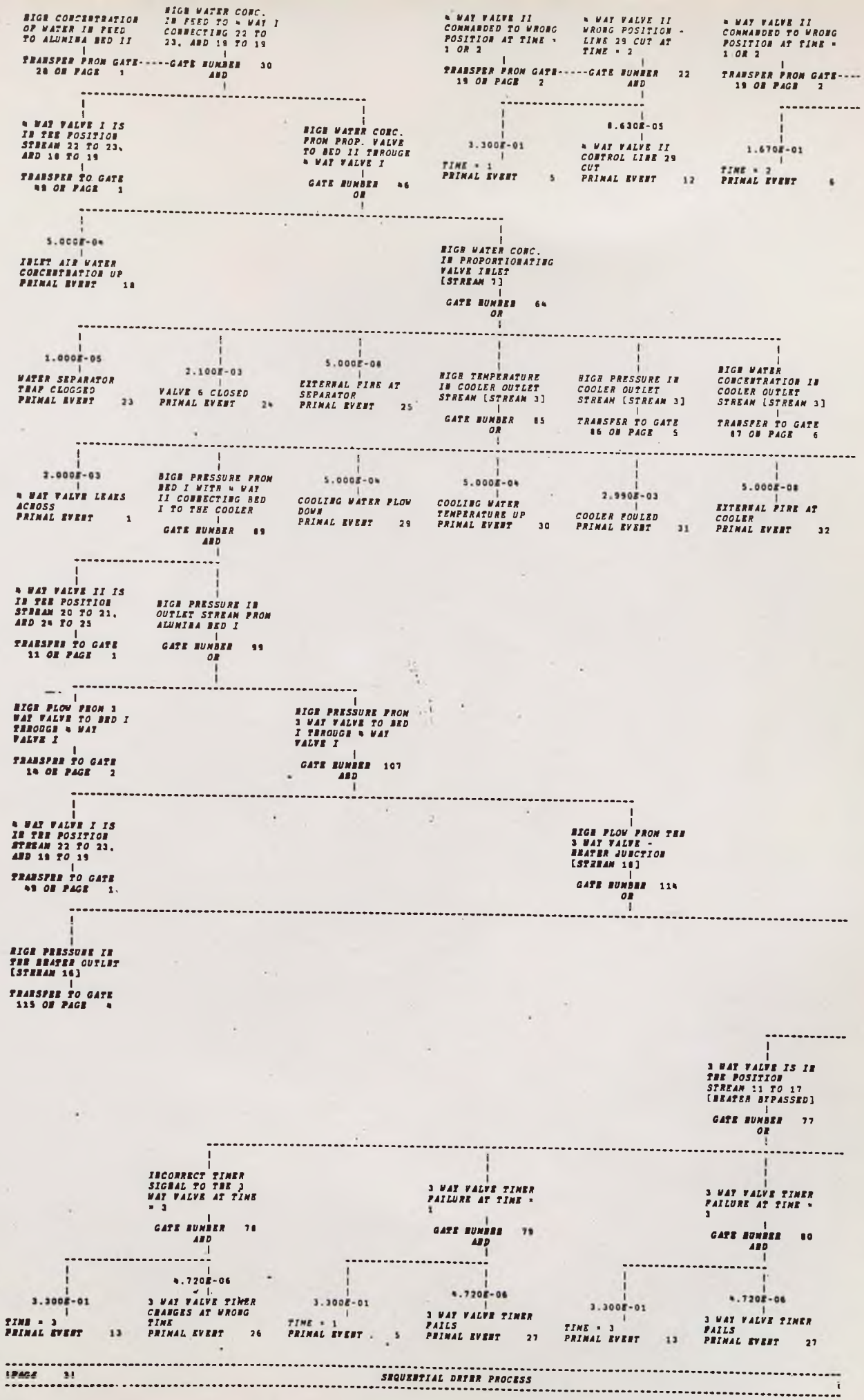


Figure 9 *continues*


HIGH CONCENTRATION OF WATER IN FEED TO ALUMINA BED II
TRANSFER FROM GATE 28 ON PAGE 1
HIGH WATER CONC. IN FEED TO 4 WAY I CONNECTING 22 TO 23, AND 18 TO 19
GATE NUMBER 30 AND
4 WAY VALVE I IS IN THE POSITION STREAM 22 TO 23, AND 18 TO 19
TRANSFER TO GATE 48 ON PAGE 1
HIGH WATER CONC. FROM PROP. VALVE TO BED II THROUGH 4 WAY VALVE I
GATE NUMBER 46 OR
5.000E-04
INLET AIR WATER CONCENTRATION UP PRIMAL EVENT 18
HIGH WATER CONC. IN PROPORTIONATING VALVE INLET [STREAM 7]
GATE NUMBER 64 OR
1.000E-05
WATER SEPARATOR TRAP CLOGGED PRIMAL EVENT 23
2.100E-03
VALVE 6 CLOSED PRIMAL EVENT 24
5.000E-08
EXTERNAL FIRE AT SEPARATOR PRIMAL EVENT 25
HIGH TEMPERATURE IN COOLER OUTLET STREAM [STREAM 3]
GATE NUMBER 85 OR
HIGH PRESSURE IN COOLER OUTLET STREAM [STREAM 3]
TRANSFER TO GATE 86 ON PAGE 5
HIGH WATER CONCENTRATION IN COOLER OUTLET STREAM [STREAM 3]
TRANSFER TO GATE 87 ON PAGE 6
2.000E-03
4 WAY VALVE LEAKS ACROSS PRIMAL EVENT 1
HIGH PRESSURE FROM BED I WITH 4 WAY II CONNECTING BED I TO THE COOLER
GATE NUMBER 89 AND
5.000E-04
COOLING WATER FLOW DOWN PRIMAL EVENT 29
5.000E-04
COOLING WATER TEMPERATURE UP PRIMAL EVENT 30
2.990E-03
COOLER FOULED PRIMAL EVENT 31
5.000E-08
EXTERNAL FIRE AT COOLER PRIMAL EVENT 32
4 WAY VALVE II IS IN THE POSITION STREAM 20 TO 21, AND 24 TO 25
TRANSFER TO GATE 11 ON PAGE 1
HIGH PRESSURE IN OUTLET STREAM FROM ALUMINA BED I
GATE NUMBER 99 OR
HIGH FLOW FROM 3 WAY VALVE TO BED I THROUGH 4 WAY VALVE I
TRANSFER TO GATE 14 ON PAGE 2
HIGH PRESSURE FROM 3 WAY VALVE TO BED I THROUGH 4 WAY VALVE I
GATE NUMBER 107 AND
4 WAY VALVE I IS IN THE POSITION STREAM 22 TO 23, AND 18 TO 19
TRANSFER TO GATE 48 ON PAGE 1
HIGH FLOW FROM THE 3 WAY VALVE - HEATER JUNCTION [STREAM 16]
GATE NUMBER 114 OR
HIGH PRESSURE IN THE HEATER OUTLET [STREAM 16]
TRANSFER TO GATE 115 ON PAGE 4
4 WAY VALVE II COMMANDED TO WRONG POSITION AT TIME = 1 OR 2
TRANSFER FROM GATE 19 ON PAGE 2
4 WAY VALVE II WRONG POSITION - LINE 29 CUT AT TIME = 2
GATE NUMBER 22 AND
4 WAY VALVE II COMMANDED TO WRONG POSITION AT TIME = 1 OR 2
TRANSFER FROM GATE 19 ON PAGE 2
3.300E-01
TIME = 1 PRIMAL EVENT 5
8.630E-05
4 WAY VALVE II CONTROL LINE 29 CUT PRIMAL EVENT 12
1.670E-01
TIME = 2 PRIMAL EVENT 6
3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 [HEATER BYPASSED]
GATE NUMBER 77 OR
INCORRECT TIMER SIGNAL TO THE 3 WAY VALVE AT TIME = 3
GATE NUMBER 78 AND
3 WAY VALVE TIMER FAILURE AT TIME = 1
GATE NUMBER 79 AND
3 WAY VALVE TIMER FAILURE AT TIME = 3
GATE NUMBER 80 AND
3.300E-01
TIME = 3 PRIMAL EVENT 13
4.720E-06
3 WAY VALVE TIMER CHANGES AT WRONG TIME PRIMAL EVENT 26
3.300E-01
TIME = 1 PRIMAL EVENT 5
4.720E-06
3 WAY VALVE TIMER FAILS PRIMAL EVENT 27
3.300E-01
TIME = 3 PRIMAL EVENT 13
4.720E-06
3 WAY VALVE TIMER FAILS PRIMAL EVENT 27
!PAGE 31!
SEQUENTIAL DRIER PROCESS

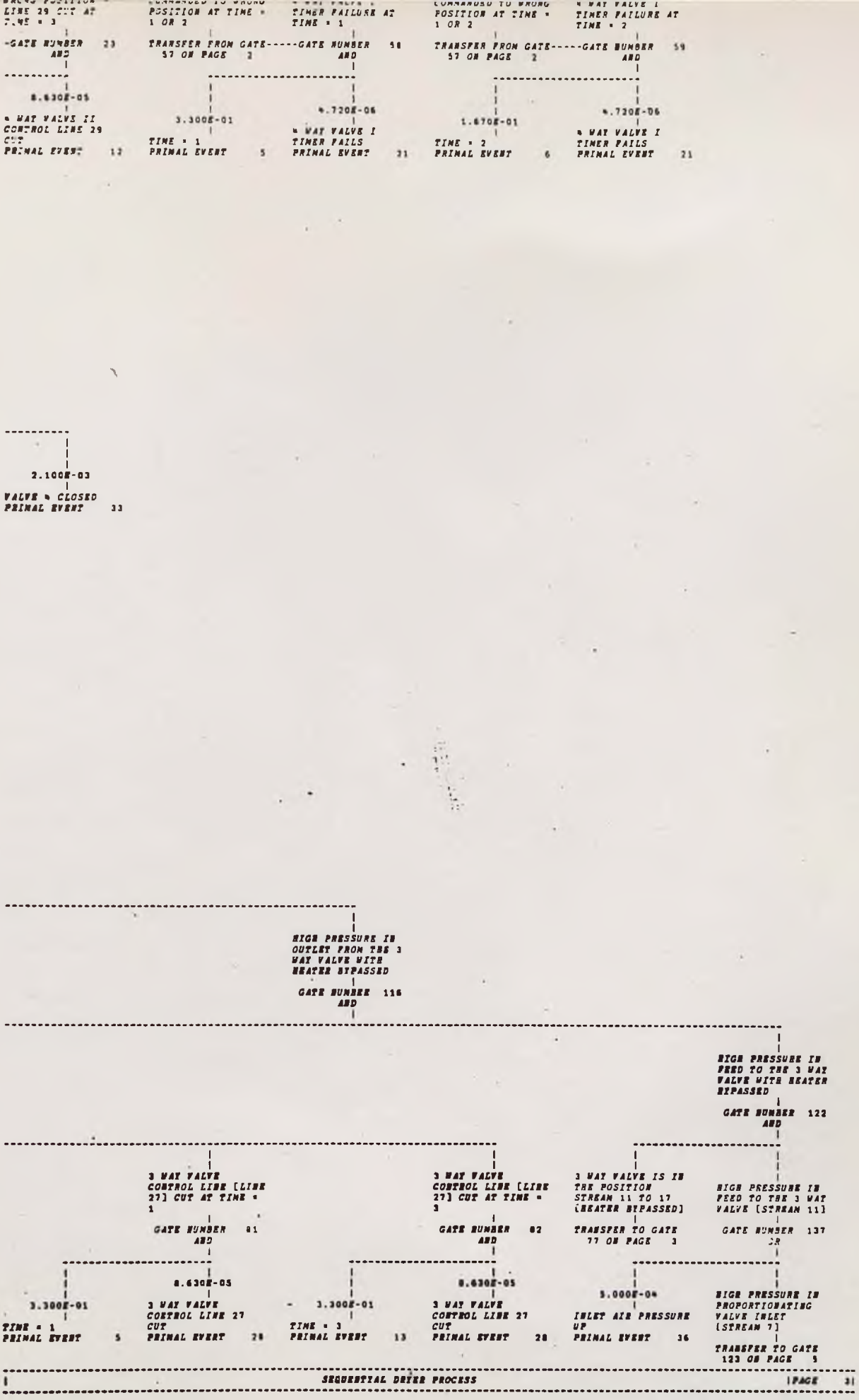
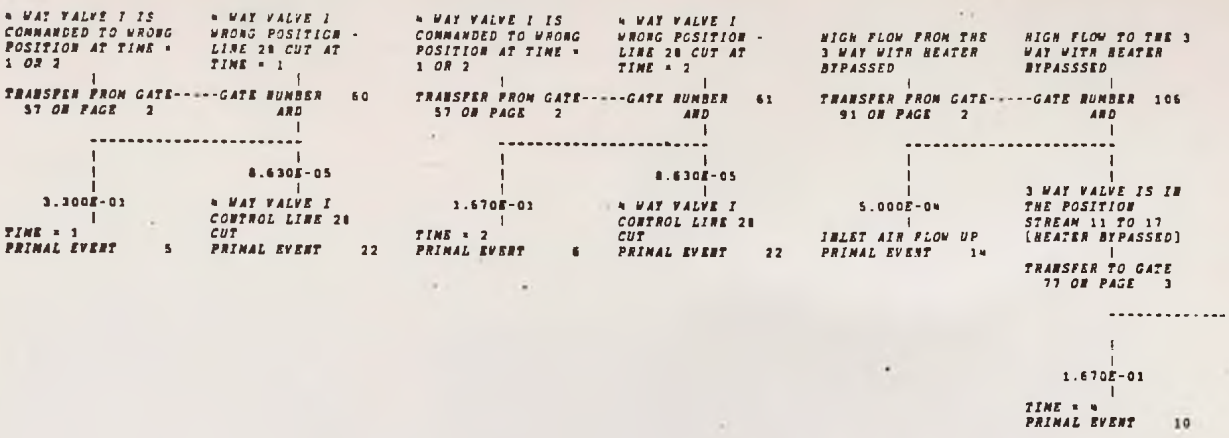
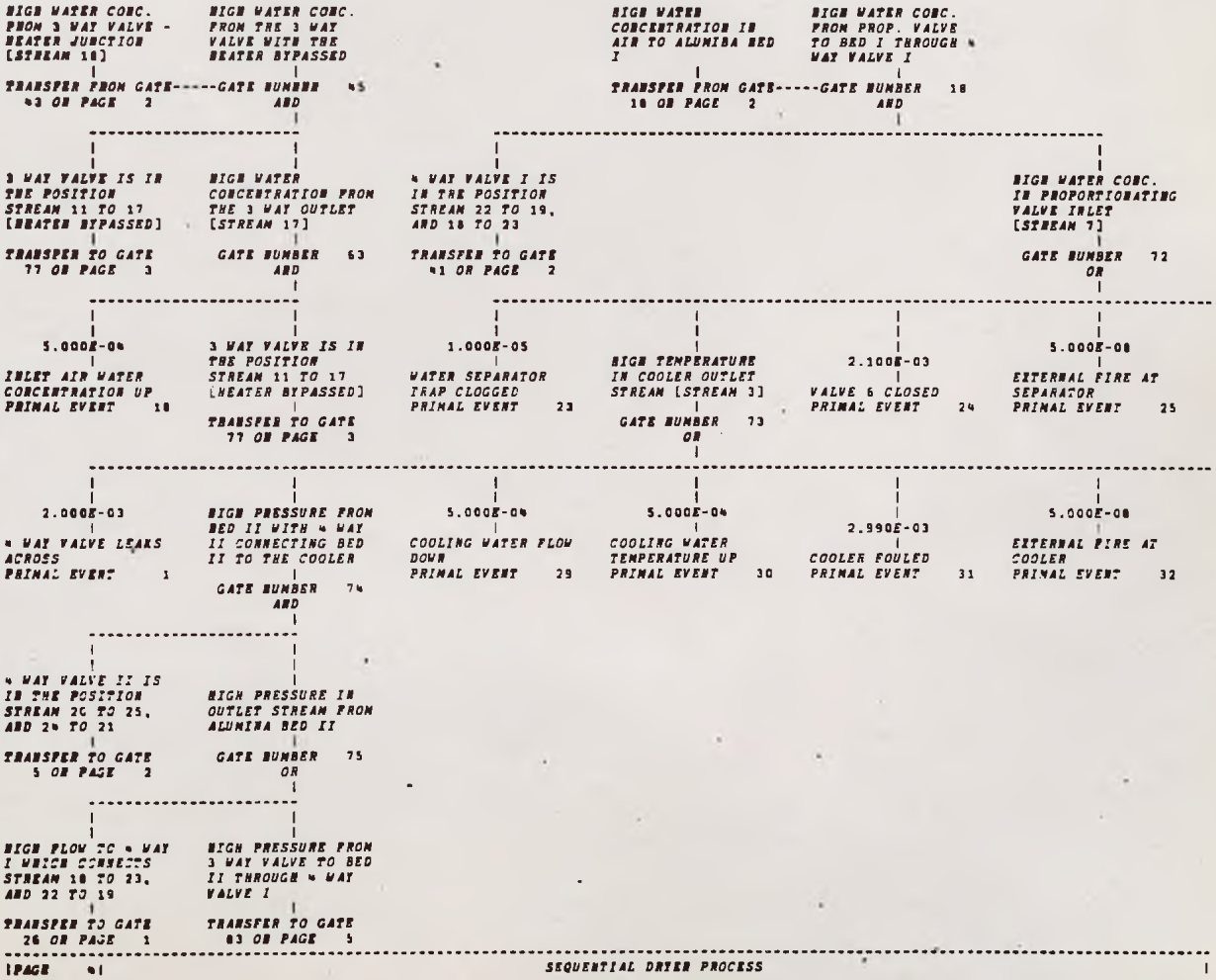
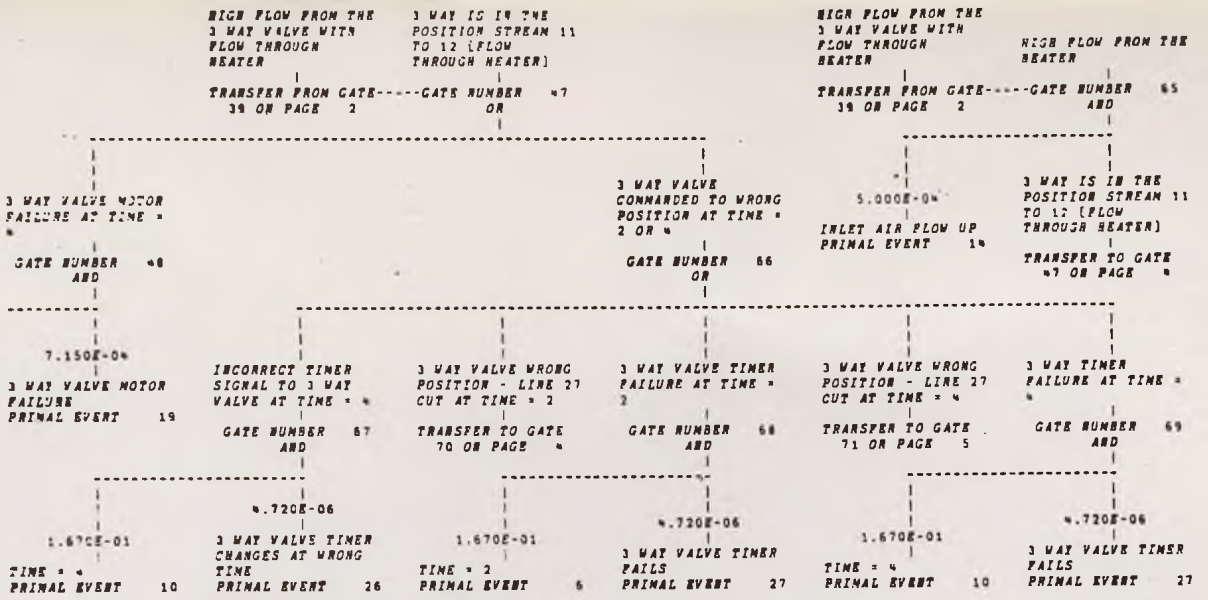
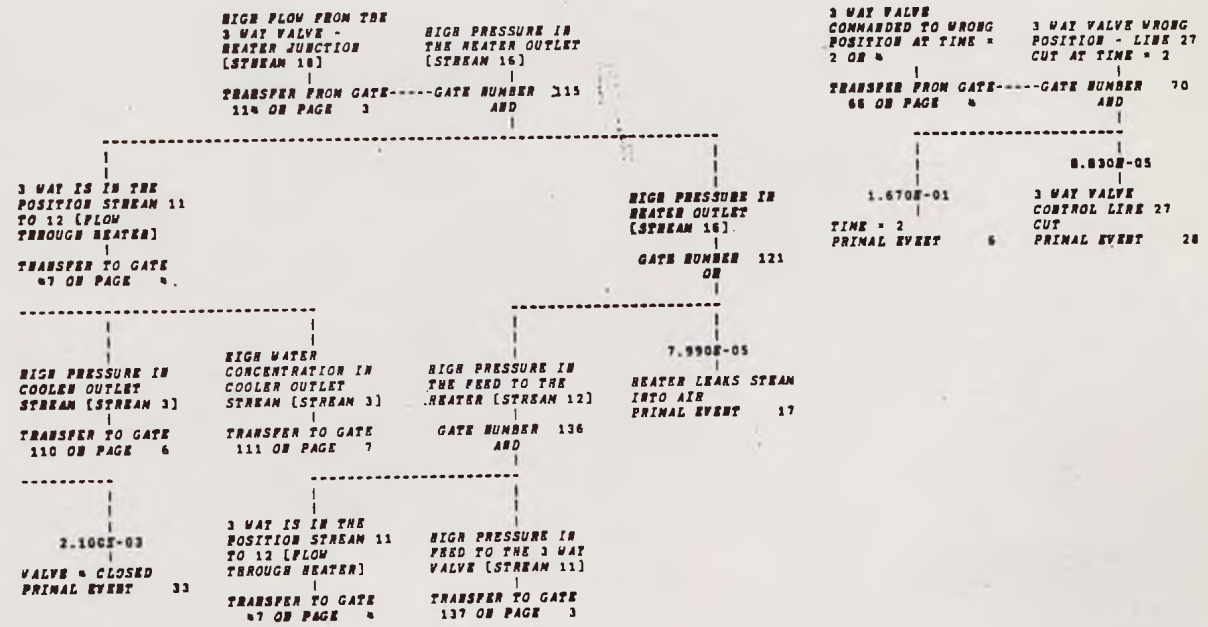


Figure 9 *continues*


4 WAY VALVE I IS COMMANDED TO WRONG POSITION AT TIME = 1 OR 2
TRANSFER FROM GATE 57 ON PAGE 2
3.300E-01
TIME = 1
PRIMAL EVENT 5
4 WAY VALVE I WRONG POSITION - LINE 2B CUT AT TIME = 1
GATE NUMBER 60
AND
8.630E-05
4 WAY VALVE I CONTROL LINE 2B CUT
PRIMAL EVENT 22
4 WAY VALVE I IS COMMANDED TO WRONG POSITION AT TIME = 1 OR 2
TRANSFER FROM GATE 57 ON PAGE 2
1.670E-01
TIME = 2
PRIMAL EVENT 6
4 WAY VALVE I WRONG POSITION - LINE 2B CUT AT TIME = 2
GATE NUMBER 61
AND
8.630E-05
4 WAY VALVE I CONTROL LINE 2B CUT
PRIMAL EVENT 22
HIGH FLOW FROM THE 3 WAY WITH HEATER BYPASSED
TRANSFER FROM GATE 91 ON PAGE 2
5.000E-04
INLET AIR FLOW UP
PRIMAL EVENT 14
HIGH FLOW TO THE 3 WAY WITH HEATER BYPASSSED
GATE NUMBER 106
AND
3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 (HEATER BYPASSED)
TRANSFER TO GATE 77 ON PAGE 3
1.670E-01
TIME = 4
PRIMAL EVENT 10

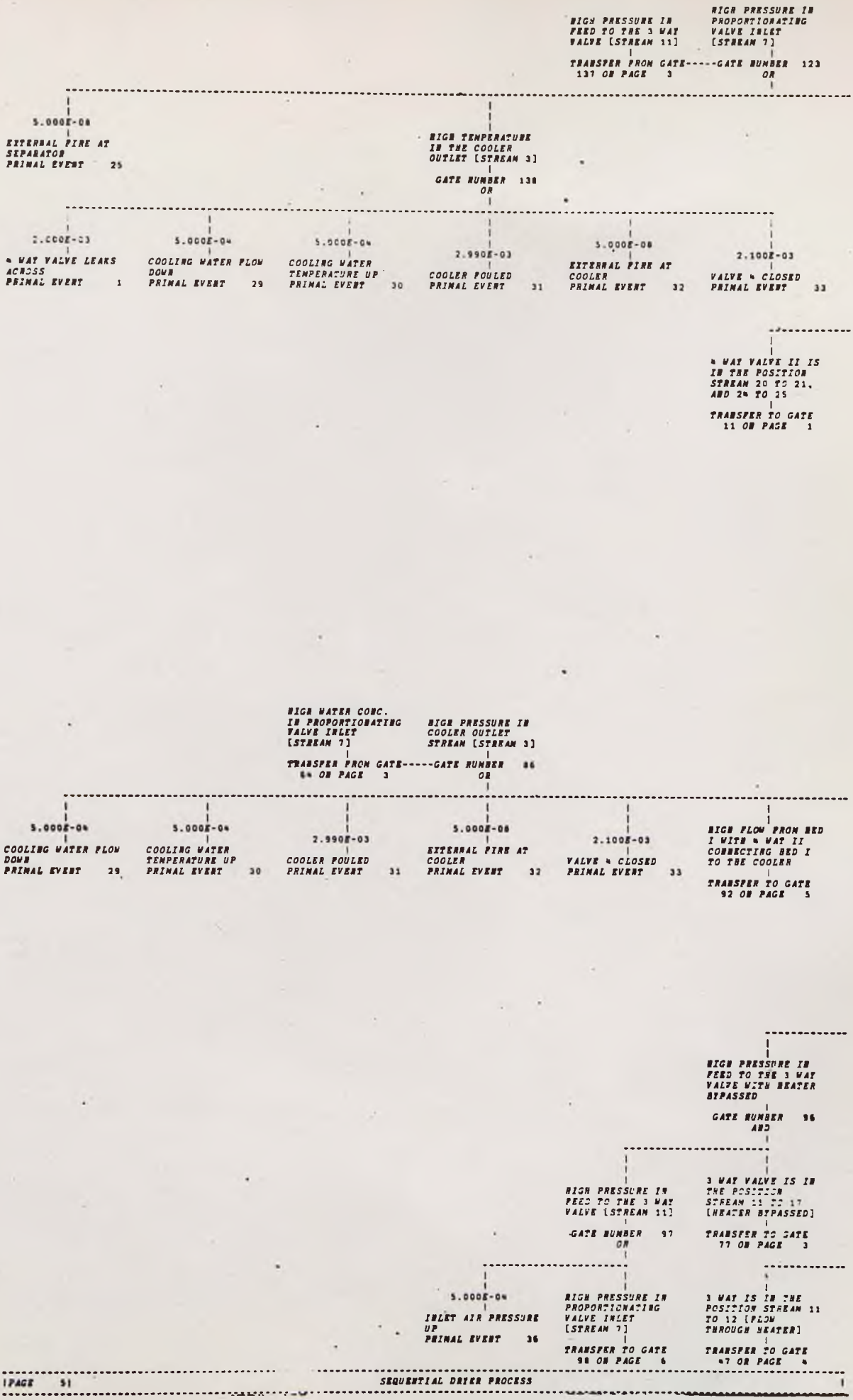

HIGH WATER CONC. FROM 3 WAY VALVE - HEATER JUNCTION (STREAM 18)
TRANSFER FROM GATE 43 ON PAGE 2
HIGH WATER CONC. FROM THE 3 WAY VALVE WITH THE HEATER BYPASSED
GATE NUMBER 45
AND
3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 (HEATER BYPASSED)
TRANSFER TO GATE 77 ON PAGE 3
HIGH WATER CONCENTRATION FROM THE 3 WAY OUTLET (STREAM 17)
GATE NUMBER 63
AND
5.000E-04
INLET AIR WATER CONCENTRATION UP
PRIMAL EVENT 18
3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 (HEATER BYPASSED)
TRANSFER TO GATE 77 ON PAGE 3
2.000E-03
4 WAY VALVE LEAKS ACROSS
PRIMAL EVENT 1
HIGH PRESSURE FROM BED II WITH 4 WAY II CONNECTING BED II TO THE COOLER
GATE NUMBER 74
AND
4 WAY VALVE II IS IN THE POSITION STREAM 20 TO 25, AND 24 TO 21
TRANSFER TO GATE 5 ON PAGE 2
HIGH PRESSURE IN OUTLET STREAM FROM ALUMINA BED II
GATE NUMBER 75
OR
HIGH FLOW TO 4 WAY I WHICH CONNECTS STREAM 18 TO 23, AND 22 TO 19
TRANSFER TO GATE 26 ON PAGE 1
HIGH PRESSURE FROM 3 WAY VALVE TO BED II THROUGH 4 WAY VALVE I
TRANSFER TO GATE 83 ON PAGE 5
HIGH WATER CONCENTRATION IN AIR TO ALUMINA BED I
TRANSFER FROM GATE 18 ON PAGE 2
HIGH WATER CONC. FROM PROP. VALVE TO BED I THROUGH 4 WAY VALVE I
GATE NUMBER 18
AND
4 WAY VALVE I IS IN THE POSITION STREAM 22 TO 19, AND 18 TO 23
TRANSFER TO GATE 41 ON PAGE 2
1.000E-05
WATER SEPARATOR TRAP CLOGGED
PRIMAL EVENT 23
HIGH TEMPERATURE IN COOLER OUTLET STREAM (STREAM 3)
GATE NUMBER 73
OR
5.000E-04
COOLING WATER FLOW DOWN
PRIMAL EVENT 29
5.000E-04
COOLING WATER TEMPERATURE UP
PRIMAL EVENT 30
2.100E-03
VALVE 6 CLOSED
PRIMAL EVENT 24
2.990E-03
COOLER FOULED
PRIMAL EVENT 31
HIGH WATER CONC. IN PROPORTIONATING VALVE INLET (STREAM 7)
GATE NUMBER 72
OR
5.000E-08
EXTERNAL FIRE AT SEPARATOR
PRIMAL EVENT 25
5.000E-08
EXTERNAL FIRE AT COOLER
PRIMAL EVENT 32





SEQUENTIAL DRIER PROCESS | PAGE %1

Figure 9 *continues*


HIGH PRESSURE IN FEED TO THE 3 WAY VALVE [STREAM 11]
HIGH PRESSURE IN PROPORTIONATING VALVE INLET [STREAM 7]
TRANSFER FROM GATE 137 OR PAGE 3
GATE NUMBER 123 OR
5.000E-04
EXTERNAL FIRE AT SEPARATOR PRIMAL EVENT 25
HIGH TEMPERATURE IN THE COOLER OUTLET [STREAM 3]
GATE NUMBER 138 OR
2.000E-03
4 WAY VALVE LEAKS ACROSS PRIMAL EVENT 1
5.000E-04
COOLING WATER FLOW DOWN PRIMAL EVENT 29
5.000E-04
COOLING WATER TEMPERATURE UP PRIMAL EVENT 30
2.990E-03
COOLER FOULED PRIMAL EVENT 31
5.000E-04
EXTERNAL FIRE AT COOLER PRIMAL EVENT 32
2.100E-03
VALVE 4 CLOSED PRIMAL EVENT 33
4 WAY VALVE II IS IN THE POSITION STREAM 20 TO 21, AND 24 TO 25
TRANSFER TO GATE 11 OR PAGE 1
HIGH WATER CONC. IN PROPORTIONATING VALVE INLET [STREAM 7]
HIGH PRESSURE IN COOLER OUTLET STREAM [STREAM 3]
TRANSFER FROM GATE 64 OR PAGE 3
GATE NUMBER 86 OR
5.000E-04
COOLING WATER FLOW DOWN PRIMAL EVENT 29
5.000E-04
COOLING WATER TEMPERATURE UP PRIMAL EVENT 30
2.990E-03
COOLER FOULED PRIMAL EVENT 31
5.000E-04
EXTERNAL FIRE AT COOLER PRIMAL EVENT 32
2.100E-03
VALVE 4 CLOSED PRIMAL EVENT 33
HIGH FLOW FROM BED I WITH 4 WAY II CONNECTING BED I TO THE COOLER
TRANSFER TO GATE 92 OR PAGE 5
HIGH PRESSURE IN FEED TO THE 3 WAY VALVE WITH HEATER BYPASSED
GATE NUMBER 96 AND
HIGH PRESSURE IN FEED TO THE 3 WAY VALVE [STREAM 11]
GATE NUMBER 97 OR
3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 [HEATER BYPASSED]
TRANSFER TO GATE 77 OR PAGE 3
5.000E-04
INLET AIR PRESSURE UP PRIMAL EVENT 36
HIGH PRESSURE IN PROPORTIONATING VALVE INLET [STREAM 7]
TRANSFER TO GATE 98 OR PAGE 6
3 WAY IS IN THE POSITION STREAM 11 TO 12 [FLOW THROUGH HEATER]
TRANSFER TO GATE 47 OR PAGE 4
PAGE 51
SEQUENTIAL DRYER PROCESS

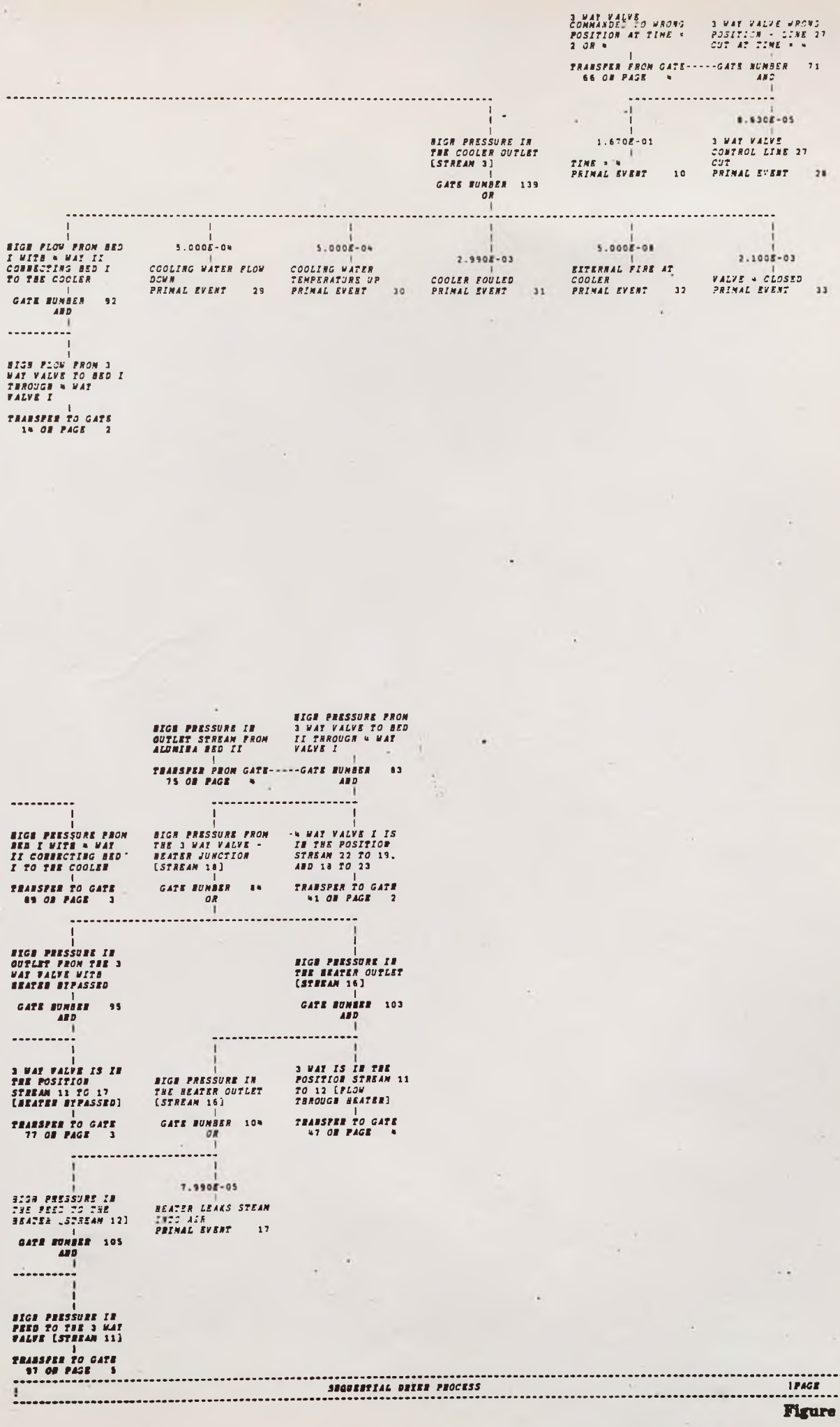
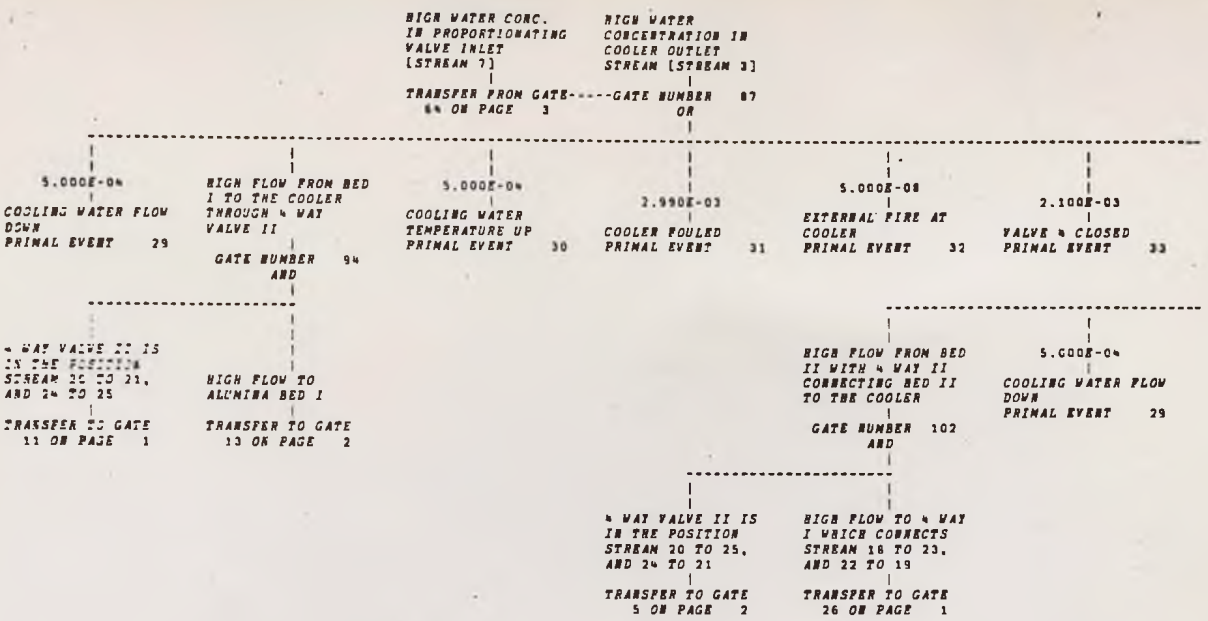
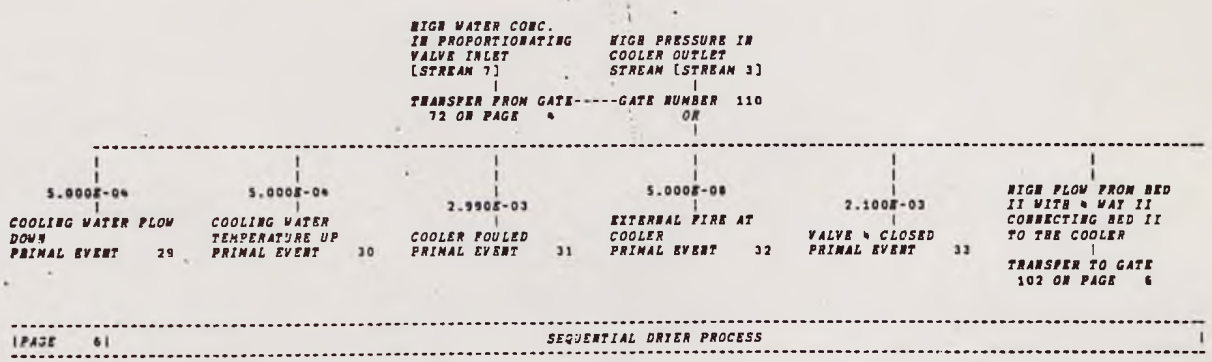
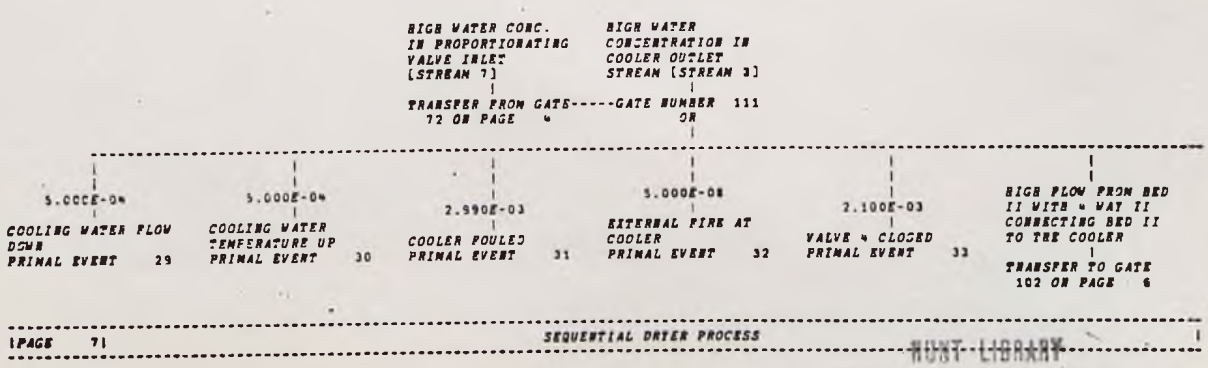
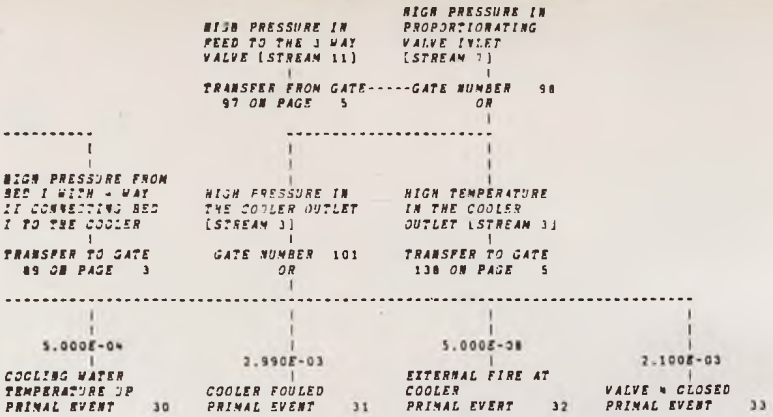


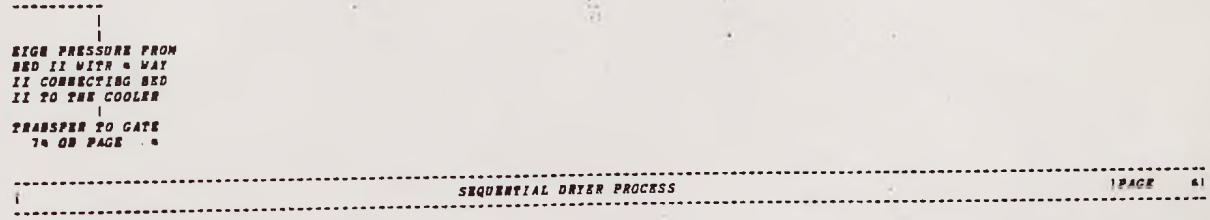
Figure 9 *continues*











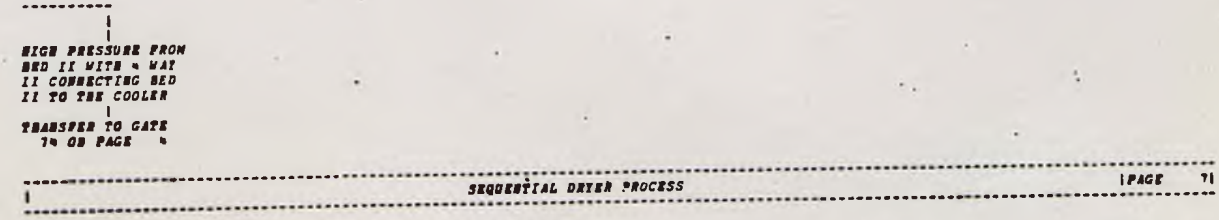


Figure 9 *continues*

GATE TABLE OF CONTENTS

TITLE: SEQUENTIAL DRYER PROCESS

GATE NUMBER	GATE TYPE	PROBABILITY	DEVELOPED ON PAGE	TEXT		
1	OR		1	HIGH WATER	CONCENTRATION IN	OUTLET AIR [STREAM 25]
2	AND		1	HIGH WATER	CONCENTRATION FROM	ALUMINA BED I
3	AND		1	HIGH WATER	CONCENTRATION FROM	ALUMINA BED II
4	OR		2	HIGH WATER	CONCENTRATION FROM	BED I IN DRYING SERVICE
5	OR		2	4 WAY VALVE II IS	IN THE POSITION	STREAM 20 TO 25, AND 24 TO 21
6	AND		2	4 WAY VALVE II	MOTOR FAILURE AT	TIME = 1
7	AND		2	4 WAY VALVE II	MOTOR FAILURE AT	TIME = 2
8	AND		2	INCORRECT SIGNAL	TO 4 WAY VALVE II	DURING TIME = 1
9	AND		2	INCORRECT SIGNAL	TO 4 WAY VALVE II	DURING TIME = 2
10	OR		1	HIGH WATER	CONCENTRATION FROM	BED II IN DRYING SERVICE
11	OR		1	4 WAY VALVE II IS	IN THE POSITION	STREAM 20 TO 21, AND 24 TO 25
12	AND		1	4 WAY VALVE II	MOTOR FAILURE AT	TIME = 4
13	OR		2	HIGH FLOW TO	ALUMINA BED I	
14	AND		2	HIGH FLOW FROM 3	WAY VALVE TO BED I	THROUGH 4 WAY VALVE I
15	AND		2	HIGH FLOW FROM	PROP. VALVE TO BED I	THROUGH 4 WAY VALVE I
16	OR		2	HIGH WATER	CONCENTRATION IN	AIR TO ALUMINA BED I
17	AND		2	HIGH WATER CONC.	FROM 3 WAY VALVE	TO BED I THROUGH 4 WAY VALVE I
18	AND		4	HIGH WATER CONC.	FROM PROP. VALVE	TO BED I THROUGH 4 WAY VALVE I
19	OR		2	4 WAY VALVE II	COMMANDED TO WRONG	POSITION AT TIME = 1 OR 2
20	AND		2	4 WAY VALVE II	TIMER FAILURE AT	TIME = 1
21	AND		2	4 WAY VALVE II	TIMER FAILURE AT	TIME = 2
22	AND		3	4 WAY VALVE II	WRONG POSITION -	LINE 29 CUT AT TIME = 2
23	AND		3	4 WAY VALVE II	WRONG POSITION -	LINE 29 CUT AT TIME = 3
25	OR		1	HIGH FLOW TO	ALUMINA BED II	
26	AND		1	HIGH FLOW TO 4 WAY I	WHICH CONNECTS	STREAM 18 TO 23, AND 22 TO 19
27	AND		1	HIGH FLOW TO 4 WAY I	WHICH CONNECTS	STREAM 22 TO 23, AND 18 TO 19
28	OR		1	HIGH CONCENTRATION OF WATER IN FEED		TO ALUMINA BED II
29	AND		1	HIGH WATER CONC.	IN FEED TO 4 WAY I	CORRECTING 18 TO 23, AND 22 TO 19
30	AND		3	HIGH WATER CONC.	IN FEED TO 4 WAY I	CORRECTING 22 TO 23, AND 18 TO 19
31	OR		1	4 WAY VALVE II	COMMANDED TO WRONG	POSITION
32	AND		1	4 WAY VALVE II	WRONG SIGNAL AT	TIME = 3
33	AND		1	4 WAY VALVE II	WRONG SIGNAL AT	TIME = 4
34	AND		1	4 WAY VALVE II	TIMER FAILURE AT	TIME = 3
35	AND		1	4 WAY VALVE II	TIMER FAILURE AT	TIME = 4
36	AND		1	4 WAY VALVE II	WRONG POSITION -	LINE 29 CUT AT TIME = 3
37	AND		1	4 WAY VALVE II	WRONG POSITION -	LINE 29 CUT AT TIME = 4
38	AND		2	HIGH FLOW FROM THE 3 WAY VALVE WITH	FLOW THROUGH	HEATER
80	OR		1	HIGH FLOW TO 4 WAY VALVE I FROM FEED		STREAM 22
81	OR		2	4 WAY VALVE I IS	IN THE POSITION	STREAM 22 TO 19, AND 18 TO 23
82	AND		2	4 WAY VALVE I	MOTOR FAILURE AT	TIME = 1
83	OR		2	HIGH WATER CONC.	FROM 3 WAY VALVE -	HEATER JUNCTION [STREAM 18]
84	AND		2	HEATER LEAKS STEAM	INTO AIR WITH 3	WAY DIRECTING FLOW THROUGH THE HEATER
85	AND		4	HIGH WATER CONC.	FROM THE 3 WAY	VALVE WITH THE HEATER BYPASSED
86	OR		3	HIGH WATER CONC.	FROM PROP. VALVE	TO BED II THROUGH 4 WAY VALVE I
87	OR		4	3 WAY IS IN THE	POSITION STREAM 11	TO 12 [FLOW THROUGH HEATER]
88	AND		4	3 WAY VALVE MOTOR	FAILURE AT TIME = 4	
89	OR		1	4 WAY VALVE I IS	IN THE POSITION	STREAM 22 TO 23, AND 18 TO 19
50	AND		1	4 WAY VALVE I	WRONG SIGNAL AT	TIME = 3
51	AND		1	4 WAY VALVE I	WRONG SIGNAL AT	TIME = 4
52	AND		1	4 WAY VALVE I	TIMER FAILURE AT	TIME = 4
53	AND		1	4 WAY VALVE I	WRONG POSITION -	LINE 28 CUT AT TIME = 3
54	AND		2	4 WAY VALVE I	WRONG POSITION -	LINE 28 CUT AT TIME = 4
56	OR		1	HIGH FLOWRATE IN	PROPORTIONATING	VALVE INLET [STREAM 7]
57	OR		2	4 WAY VALVE I IS	COMMANDED TO WRONG	POSITION AT TIME = 1 OR 2
58	AND		3	4 WAY VALVE I	TIMER FAILURE AT	TIME = 1
59	AND		3	4 WAY VALVE I	TIMER FAILURE AT	TIME = 2
60	AND		4	4 WAY VALVE I	WRONG POSITION -	LINE 28 CUT AT TIME = 1
61	AND		4	4 WAY VALVE I	WRONG POSITION -	LINE 28 CUT AT TIME = 2
63	AND		4	HIGH WATER	CONCENTRATION FROM	THE 3 WAY OUTLET [STREAM 17]
64	OR		3	HIGH WATER CONC.	IN PROPORTIONATING	VALVE INLET [STREAM 7]
65	AND		4	HIGH FLOW FROM THE HEATER		

GATE TABLE OF CONTENTS

TITLE: SEQUENTIAL DRYER PROCESS

GATE NUMBER	GATE TYPE	PROBABILITY	DEVELOPED ON PAGE	TEXT
66	OR		4	3 WAY VALVE COMMANDED TO WRONG POSITION AT TIME = 2 OR 4
67	AND		4	INCORRECT TIMER SIGNAL TO 3 WAY VALVE AT TIME = 4
68	AND		4	3 WAY VALVE TIMER FAILURE AT TIME = 2
69	AND		4	3 WAY TIMER FAILURE AT TIME = 4
70	AND		4	3 WAY VALVE WRONG POSITION - LINE 27 CUT AT TIME = 2
71	AND		5	3 WAY VALVE WRONG POSITION - LINE 27 CUT AT TIME = 4
72	OR		4	HIGH WATER CONC. IN PROPORTIONATING VALVE INLET [STREAM 7]
73	OR		4	HIGH TEMPERATURE IN COOLER OUTLET STREAM [STREAM 3]
74	AND		4	HIGH PRESSURE FROM BED II WITH 4 WAY II CONNECTING BED II TO THE COOLER
75	OR		4	HIGH PRESSURE IN OUTLET STREAM FROM ALUMINA BED II
77	OR		3	3 WAY VALVE IS IN THE POSITION STREAM 11 TO 17 [HEATER BYPASSED]
78	AND		3	INCORRECT TIMER SIGNAL TO THE 3 WAY VALVE AT TIME = 3
79	AND		3	3 WAY VALVE TIMER FAILURE AT TIME = 1
80	AND		3	3 WAY VALVE TIMER FAILURE AT TIME = 3
81	AND		3	3 WAY VALVE CONTROL LINE [LINE 27] CUT AT TIME = 1
82	AND		3	3 WAY VALVE CONTROL LINE [LINE 27] CUT AT TIME = 3
83	AND		5	HIGH PRESSURE FROM 3 WAY VALVE TO BED II THROUGH 4 WAY VALVE I
84	OR		5	HIGH PRESSURE FROM THE 3 WAY VALVE - HEATER JUNCTION [STREAM 16]
85	OR		3	HIGH TEMPERATURE IN COOLER OUTLET STREAM [STREAM 3]
86	OR		5	HIGH PRESSURE IN COOLER OUTLET STREAM [STREAM 3]
87	OR		6	HIGH WATER CONCENTRATION IN COOLER OUTLET STREAM [STREAM 3]
88	OR		2	HIGH FLOW FROM THE 3 WAY VALVE - HEATER JUNCTION [STREAM 16]
89	AND		3	HIGH PRESSURE FROM BED I WITH 4 WAY II CONNECTING BED I TO THE COOLER
91	AND		2	HIGH FLOW FROM THE 3 WAY WITH HEATER BYPASSED
92	AND		5	HIGH FLOW FROM BED I WITH 4 WAY II CONNECTING BED I TO THE COOLER
94	AND		6	HIGH FLOW FROM BED I TO THE COOLER THROUGH 4 WAY VALVE II
95	AND		5	HIGH PRESSURE IN OUTLET FROM THE 3 WAY VALVE WITH HEATER BYPASSED
96	AND		5	HIGH PRESSURE IN FEED TO THE 3 WAY VALVE WITH HEATER BYPASSED
97	OR		5	HIGH PRESSURE IN FEED TO THE 3 WAY VALVE [STREAM 11]
98	OR		6	HIGH PRESSURE IN PROPORTIONATING VALVE INLET [STREAM 7]
99	OR		3	HIGH PRESSURE IN OUTLET STREAM FROM ALUMINA BED I
101	OR		6	HIGH PRESSURE IN THE COOLER OUTLET [STREAM 3]
102	AND		6	HIGH FLOW FROM BED II WITH 4 WAY II CONNECTING BED II TO THE COOLER
103	AND		5	HIGH PRESSURE IN THE HEATER OUTLET [STREAM 16]
104	OR		5	HIGH PRESSURE IN THE HEATER OUTLET [STREAM 16]
105	AND		5	HIGH PRESSURE IN THE FEED TO THE HEATER [STREAM 12]
106	AND		4	HIGH FLOW TO THE 3 WAY WITH HEATER BYPASSSED
107	AND		3	HIGH PRESSURE FROM 3 WAY VALVE TO BED I THROUGH 4 WAY VALVE I
110	OR		6	HIGH PRESSURE IN COOLER OUTLET STREAM [STREAM 3]
111	OR		7	HIGH WATER CONCENTRATION IN COOLER OUTLET STREAM [STREAM 3]
114	OR		3	HIGH FLOW FROM THE 3 WAY VALVE - HEATER JUNCTION [STREAM 16]
115	AND		4	HIGH PRESSURE IN THE HEATER OUTLET [STREAM 16]
116	AND		3	HIGH PRESSURE IN OUTLET FROM THE 3 WAY VALVE WITH HEATER BYPASSED
121	OR		4	HIGH PRESSURE IN HEATER OUTLET [STREAM 16]
122	AND		3	HIGH PRESSURE IN FEED TO THE 3 WAY VALVE WITH HEATER BYPASSED
123	OR		5	HIGH PRESSURE IN PROPORTIONATING VALVE INLET [STREAM 7]
136	AND		4	HIGH PRESSURE IN THE FEED TO THE HEATER [STREAM 12]
137	OR		3	HIGH PRESSURE IN FEED TO THE 3 WAY VALVE [STREAM 11]
138	OR		5	HIGH TEMPERATURE IN THE COOLER OUTLET [STREAM 3]
139	OR		5	HIGH PRESSURE IN THE COOLER OUTLET [STREAM 3]

Figure 9 *continues*

EVENT TABLE OF CONTENTS

TITLE: SEQUENTIAL DRYER PROCESS

EVENT NUMBER	PROBABILITY	DEVELOPED ON PAGE	TEXT
1	2.000E-03	1	4 WAY VALVE LEAKS ACROSS
2	5.000E-08	2	FIRE AT BED I
3	1.200E-04	2	NO ALUMINA IN BED I, OR CHANNELING
4	7.150E-04	1	4 WAY VALVE II MOTOR FAILURE
5	3.300E-01	2	TIME = 1
6	1.670E-01	2	TIME = 2
7	4.720E-06	1	4 WAY VALVE II TIMER CHANGES AT WRONG TIME
8	5.000E-08	1	FIRE AT BED II
9	1.200E-04	1	NO ALUMINA IN BED II, OR CHANNELING
10	1.670E-01	1	TIME = 4
11	4.720E-06	1	4 WAY VALVE II TIMER FAILS
12	6.630E-05	1	4 WAY VALVE II CONTROL LINE 29 CUT
13	3.300E-01	1	TIME = 3
14	5.000E-04	1	INLET AIR FLOW UP
15	7.150E-04	2	4 WAY VALVE I MOTOR FAILURE
17	7.990E-05	2	HEATER LEAKS STEAM INTO AIR
18	5.000E-04	3	INLET AIR WATER CONCENTRATION UP
19	7.150E-04	4	3 WAY VALVE MOTOR FAILURE
20	4.720E-06	1	4 WAY VALVE I TIMER CHANGES AT WRONG TIME
21	4.720E-06	1	4 WAY VALVE I TIMER FAILS
22	6.630E-05	1	4 WAY VALVE I CONTROL LINE 28 CUT
23	1.000E-05	1	WATER SEPARATOR TRAP CLOGGED
24	2.100E-03	1	VALVE 6 CLOSED
25	5.000E-08	3	EXTERNAL FIRE AT SEPARATOR
26	4.720E-06	3	3 WAY VALVE TIMER CHANGES AT WRONG TIME
27	4.720E-06	3	3 WAY VALVE TIMER FAILS
28	6.630E-05	3	3 WAY VALVE CONTROL LINE 27 CUT
29	5.000E-04	3	COOLING WATER FLOW DOWN
30	5.000E-04	3	COOLING WATER TEMPERATURE UP
31	2.990E-03	3	COOLER FOULED
32	5.000E-08	3	EXTERNAL FIRE AT COOLER
33	2.100E-03	3	VALVE 5 CLOSED
34	5.000E-04	3	INLET AIR PRESSURE UP

Figure 9. The fault tree for the event water concentration too high from the utility air dryer process.

the structure of the digraph. It is not necessary for the analyst to preordain the logic (AND, OR, etc.) of the interactions between variables.

With this type of model all foreseeable interactions may be described. What remains is to interconnect the digraph model for each piece of equipment in the system to obtain a model for the complete system. Figure 4 shows a partial digraph for the heat exchanger system shown in Figure 1. From the digraph model, fault trees may be directly deduced. The algorithm for this deduction has been described previously (Lapp, 1977). Briefly, the procedure involves starting at the node in the digraph which denotes the top event. The negative feedforward and feedback loops through a node determine how it should be logically related to its inputs. The algorithm has over 30 different logical expansions of a node. The input nodes are logically expanded in a similar manner until the complete fault tree is obtained. The consistency of intermediate events and variables is maintained during the generation of the fault tree.

After generation of the fault tree, the tree is listed, "drawn" on a line printer, and put in minimal cut-set form. The minimal cut-sets of a Boolean equation are the sets of events which are sufficient to cause the top event and do not contain any other sufficient sets of events. The fault tree for the event temperature in stream 4 (T4) too high is given in Figure 5.

The following example illustrates the application of this strategy to a sequential process for drying air.

Example: Fixed Bed Alumina Air Dryers. Figure 6 illustrates a process for drying air. Ambient air which contains water vapor enters in stream 9. The air passes through a bed of alumina (Bed I) where the water vapor is adsorbed. The dried air passes out of the process in stream 25. This process has been used by Professor C. J. King of the Department of Chemical Engineering, University of California, Berkeley, Calif., as a case study in process design.

In order to maintain a continuous supply of dry air, two beds of alumina are employed. When one bed is removing water from the inlet air, the other bed is being regenerated. Regeneration involves passing hot air through a bed which has been loaded to capacity with water. The hot air strips the water from the alumina. The hot air leaving the regenerating bed is passed through a condenser where water is removed. The air is reheated and passed through the operating dryer. The regenerated bed is then cooled with inlet air and switched back into service. The same procedure is followed for the other bed. Table I gives the sequence of operations for a complete cycle.

If the outlet air from the process contains too much water a number of pieces of valuable equipment downstream may be destroyed. What could cause the water concentration in stream 25 to be too high? One way to answer this question is to construct a fault tree for the event concentration of water too high in stream 25 (C (+1) Stream 25).

Input–output models for several of the pieces of equipment

```
PROBLEM ID:  SEQUENTIAL DRYER PROCESS

          162 MINIMAL CUT SETS GENERATED.

          TOP EVENT PROBABILITY= 2.0001E-03

MINIMAL CUT SET NO.      1        ITS PROBABILITY:   2.00E-03
 EVENT      1( 2.00E-03) 4 WAY VALVE LEAKS ACROSS

MINIMAL CUT SET NO.      2        ITS PROBABILITY:   2.83E-08
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      5( 3.30E-01) TIME = 1

MINIMAL CUT SET NO.      3        ITS PROBABILITY:   1.43E-08
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      6( 1.67E-01) TIME = 2

MINIMAL CUT SET NO.      4        ITS PROBABILITY:   1.43E-08
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     10( 1.67E-01) TIME = 4

MINIMAL CUT SET NO.      5        ITS PROBABILITY:   3.42E-09
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     12( 8.63E-05) 4 WAY VALVE II CONTROL LINE 29 CUT

MINIMAL CUT SET NO.      6        ITS PROBABILITY:   3.42E-09
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     12( 8.63E-05) 4 WAY VALVE II CONTROL LINE 29 CUT
 EVENT     13( 3.30E-01) TIME = 3

MINIMAL CUT SET NO.      7        ITS PROBABILITY:   1.73E-09
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     10( 1.67E-01) TIME = 4
 EVENT     12( 8.63E-05) 4 WAY VALVE II CONTROL LINE 29 CUT

MINIMAL CUT SET NO.      8        ITS PROBABILITY:   1.73E-09
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      6( 1.67E-01) TIME = 2
 EVENT     12( 8.63E-05) 4 WAY VALVE II CONTROL LINE 29 CUT

MINIMAL CUT SET NO.      9        ITS PROBABILITY:   5.04E-10
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     16( 7.15E-04) 4 WAY VALVE I MOTOR FAILURE
 EVENT     31( 2.99E-03) COOLER FOULED

MINIMAL CUT SET NO.     10        ITS PROBABILITY:   3.54E-10
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     16( 7.15E-04) 4 WAY VALVE I MOTOR FAILURE
 EVENT     24( 2.10E-03) VALVE 6 CLOSED

MINIMAL CUT SET NO.     11        ITS PROBABILITY:   3.54E-10
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     16( 7.15E-04) 4 WAY VALVE I MOTOR FAILURE
 EVENT     33( 2.10E-03) VALVE 4 CLOSED

MINIMAL CUT SET NO.     12        ITS PROBABILITY:   1.87E-10
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     11( 4.72E-06) 4 WAY VALVE II TIMER FAILS

MINIMAL CUT SET NO.     13        ITS PROBABILITY:   1.87E-10
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      5( 3.30E-01) TIME = 1
 EVENT      7( 4.72E-06) 4WAY VALVE II TIMER CHANGES AT WRONG
                          TIME

MINIMAL CUT SET NO.     14        ITS PROBABILITY:   1.87E-10
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     11( 4.72E-06) 4 WAY VALVE II TIMER FAILS
 EVENT     13( 3.30E-01) TIME = 3

MINIMAL CUT SET NO.     15        ITS PROBABILITY:   1.87E-10
 EVENT      7( 4.72E-06) 4WAY VALVE II TIMER CHANGES AT WRONG
                          TIME
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     13( 3.30E-01) TIME = 3

MINIMAL CUT SET NO.     16        ITS PROBABILITY:   9.46E-11
 EVENT      7( 4.72E-06) 4WAY VALVE II TIMER CHANGES AT WRONG
                          TIME
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     10( 1.67E-01) TIME = 4

MINIMAL CUT SET NO.     17        ITS PROBABILITY:   9.46E-11
 EVENT      9( 1.20E-04) NO ALUMINA IN BED II, OR CHANNELING
 EVENT     10( 1.67E-01) TIME = 4
 EVENT     11( 4.72E-06) 4 WAY VALVE II TIMER FAILS

MINIMAL CUT SET NO.     18        ITS PROBABILITY:   9.46E-11
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      6( 1.67E-01) TIME = 2
 EVENT     11( 4.72E-06) 4 WAY VALVE II TIMER FAILS

MINIMAL CUT SET NO.     19        ITS PROBABILITY:   9.46E-11
 EVENT      3( 1.20E-04) NO ALUMINA IN BED I, OR CHANNELING
 EVENT      6( 1.67E-01) TIME = 2
 EVENT      7( 4.72E-06) 4WAY VALVE II TIMER CHANGES AT WRONG
                          TIME

MINIMAL CUT SET NO.     20        ITS PROBABILITY:   8.44E-11
 EVENT      4( 7.15E-04) 4 WAY VALVE II MOTOR FAILURE
 EVENT      5( 3.30E-01) TIME = 1
 EVENT     16( 7.15E-04) 4 WAY VALVE I MOTOR FAILURE
 EVENT     30( 5.00E-04) COOLING WATER TEMPERATURE UP
```

in the system are given in Figure 7. Note the time dependent nature of the three-way valve, four-way valves, and the timer. The input–output models were interconnected to give a digraph model for the dryer system. The complete digraph for this system contained 67 nodes and 439 edges. A reduced version of the digraph is shown in Figure 8. Only the main concentration and flow interactions are shown.

The fault tree generation algorithm required 30 s of IBM 360/67 time to generate the fault tree for this system. The tree contains 143 gates and is shown in Figure 9. This tree is different from the usual fault tree in that common events such as time periods are considered.

Probability data were gathered and estimated for the events included in the tree. Table II presents the data. Over 100 cut-sets were computed for the tree. The first twenty are presented in Table III.

An analysis of the cut-sets for this system indicates the importance of leaking of the four-way valve. The results of this fault tree analysis in conjunction with economic considerations of the dryer operation and other possible design or maintenance corrections can be used to decide an appropriate action.

Conclusions

With digraph models that contain edges that depend on other variables and events, it is possible to include common sequential behavior in a system digraph. This allows the generation of a fault tree that contains events (like the sequence of valve operations) that are normally true. The analysis of the minimal cut-sets that result from this fault tree allows the analyst to focus attention on the important parts of the system.

Literature Cited

Esary, J. D., Ziehms, H., "Reliability Analysis of Phased Missions," SIAM, "Reliability and Fault Tree Analysis," 1975.

Fussell, J. B., Barlow, R. E., Ed., SIAM, "Reliability and Fault Tree Analysis," 1975.

Fussell, J. B., Powers, G. J., Bennetts, R. B., *IEEE Trans. Reliab.*, **R-23**, 1 (Apr 1974).

Lapp, S. A., Powers, G. J., *IEEE Trans. Reliab.*, (Apr 1977).

Powers, G. J., Tompkins, F. C., *AIChE J.*, **20**, 91 (1974).

Received for review September 20, 1976
Accepted April 29, 1977

APPENDIX B

Reprinted from CHEMICAL ENGINEERING PROGRESS, April 1976

PROCESS CONTROL PRACTICES

Computer-Aided Fault Tree Synthesis

Procedures and computer programs now being developed should reduce the amount of time, now measured in man-years, for carrying out a fault tree analysis.

G. J. Powers and S. A. Lapp, Carnegie-Mellon Univ., Pittsburgh, Pa.

Fault tree analysis is one means for systematically identifying cause-and-effect chains of events which could lead to environmental hazards. Once a fault tree has been constructed it is possible to compute the time dependent probability of occurrence of the hazard, given the probability of occurrence of causing events.

The two major problems with this approach are in 1) generating the tree and 2) gathering the appropriate probability data. We have developed a computer program which aids in the generation of fault trees for chemical processes. It uses signed digraph models for equipment. The fault tree is deduced directly from this simple model of the system.

Environmental risk assessment is becoming a more important aspect of the design of chemical processing plants. Environmental impact statements, potential fines, and lawsuits make the potential environmental risks an important business issue. In the assessment of risks it is necessary to determine 1) the consequences of each potential risk, 2) the probability of occurrence of each event, 3) the chains of events which could cause the risk, and 4) the costs of potential changes to the process and its operation which might reduce the probability or consequences of the event.

A number of techniques advocated for determining the consequences of risks are commonly based on simulations of the transport and interaction of released chemicals with people and the environment. The results of the simulation are estimates of the potential damages that might result from releases of various sizes and types. The adequacy of these estimates often depends on the state of knowledge of the reactivity of the chemicals. At low concentration levels of exposure the prediction of consequences is very uncertain. In the following discussion only high concentration exposures, which usually occur over short time periods, are considered.

Prediction of the probabilities of exposure is a more difficult issue. Intuitive techniques are less suited for the prediction of probabilities than they are for consequence estimation.

"Safety First," a motto used by many chemical companies, illustrates the importance industry places on preventing personal, equipment, and business interruption hazards. The employee safety records of the major chemical companies have been good. The probability of being injured in a chemical processing plant is less than in many other industries and much less than staying at home. However, the process loss situation has not been as encouraging. A number of major losses have occurred due to fires, explosions, and releases of chemicals in chemical plants. In addition, more stringent laws are being promulgated to ensure worker safety and to prevent releases of toxic or otherwise hazardous materials into the environment.

How can the chemical industry improve the safety and reliability of their processes? How can they counteract well-meaning but sometimes misdirected governmental agencies? The key lies in careful attention to the design and operation of each part of the chemical processes.

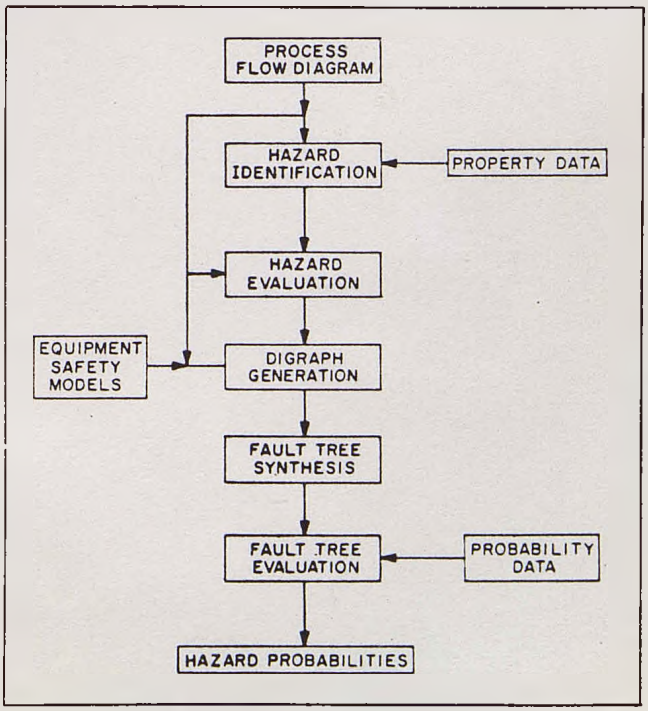


Figure 1. Steps in hazard assessment.

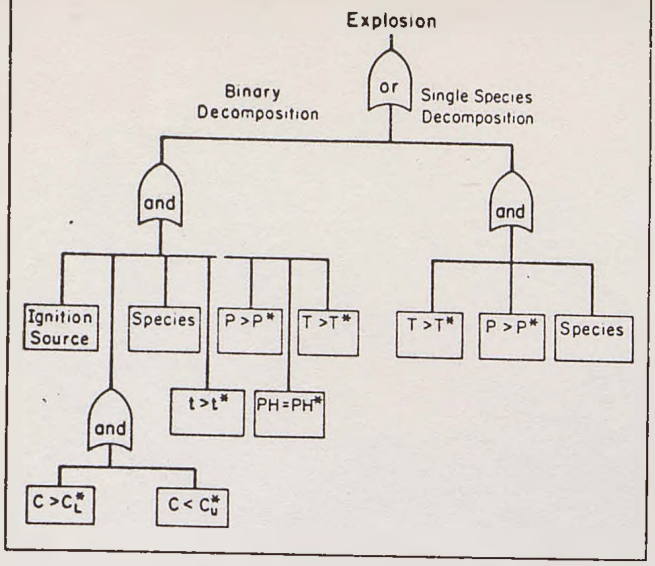


Figure 2.
The hazard equation for explosion.

Designers must be able to analyze processes to foresee potential failures. Operators of processes, as well as the automatic control systems, must be trained to respond rapidly to process failures.

In dealing with governmental agencies process designers must show that "all *reasonable* precautions have been taken to reduce the probability of events to *acceptable* levels." This last sentence contains the key features of the safety problem. The law is written so that the industry is responsible for 1) identifying the potential events, 2) deciding on *reasonable* precautions, and 3) reducing the probability of these events to *acceptable* levels. These three areas are where meaningful contributions to the safety analysis of chemical processes can be made. One needs to 1) identify events, 2) decide how to prevent these events from occurring, and 3) compute and determine what constitutes acceptable probabilities.

The following describes one approach to this analysis problem. The fundamental concepts for this approach have been described in previous papers. (*1,2*) This article will briefly describe the FTS (Fault Tree Synthesis) program and illustrate by a simple example how it might be used. The major points to be covered are 1) hazard identification, 2) hazard consequence estimation, 3) digraph models of individual equipment and complete chemical processes, 4) fault tree synthesis, 5) fault tree evaluation (probability calculations) and 6) use of the program for new designs and for the review of existing processes.

First need is to identify process flow

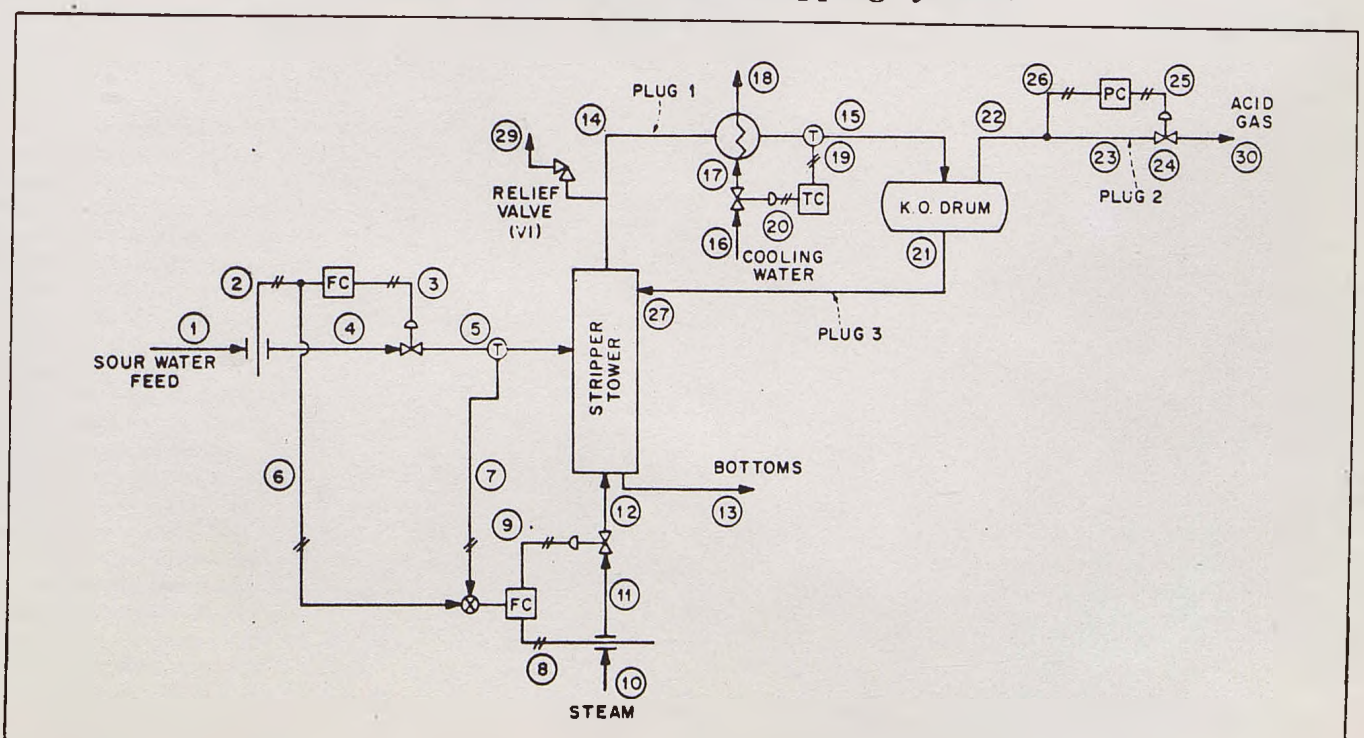
Figure 1 illustrates the major steps required to carry out the quantitative safety analysis of a chemical process. Initially the process flow diagram must be identified. Second, the hazardous events that might occur within or around the process must be discovered. Data are required on the physical and chemical properties of the species in the process and the mechanical and electrical properties of the process equipment. Each potentially hazardous event may then be evaluated to determine its possible costs (loss of life, loss of equipment and material, loss of business, and damage of the environment).

With the information derived from these steps a ranked list of potential hazards may be constructed. What is required is the probability of occurrence of each hazard.

Since most of these events will occur infrequently, it is not possible to take a direct statistical approach to the estimation of their rate of occurrence. What is needed is a means for estimating the probability of hazardous events from the probabilities of more common (and hence more accurately determined) events.

Fault tree analysis (FTA) is one means for carrying out this estimation. FTA is based on the assumption that the hazard is a logical consequence of other events. That is, if we knew the probability of occurrence of the set of events which contains the necessary precursors to a fire we could estimate the probability of the fire. In this manner, the causative events are reduced to smaller and smaller sets of events until a level is reached at which sufficient probability data are available.

Figure 3. A sour water stripping system.



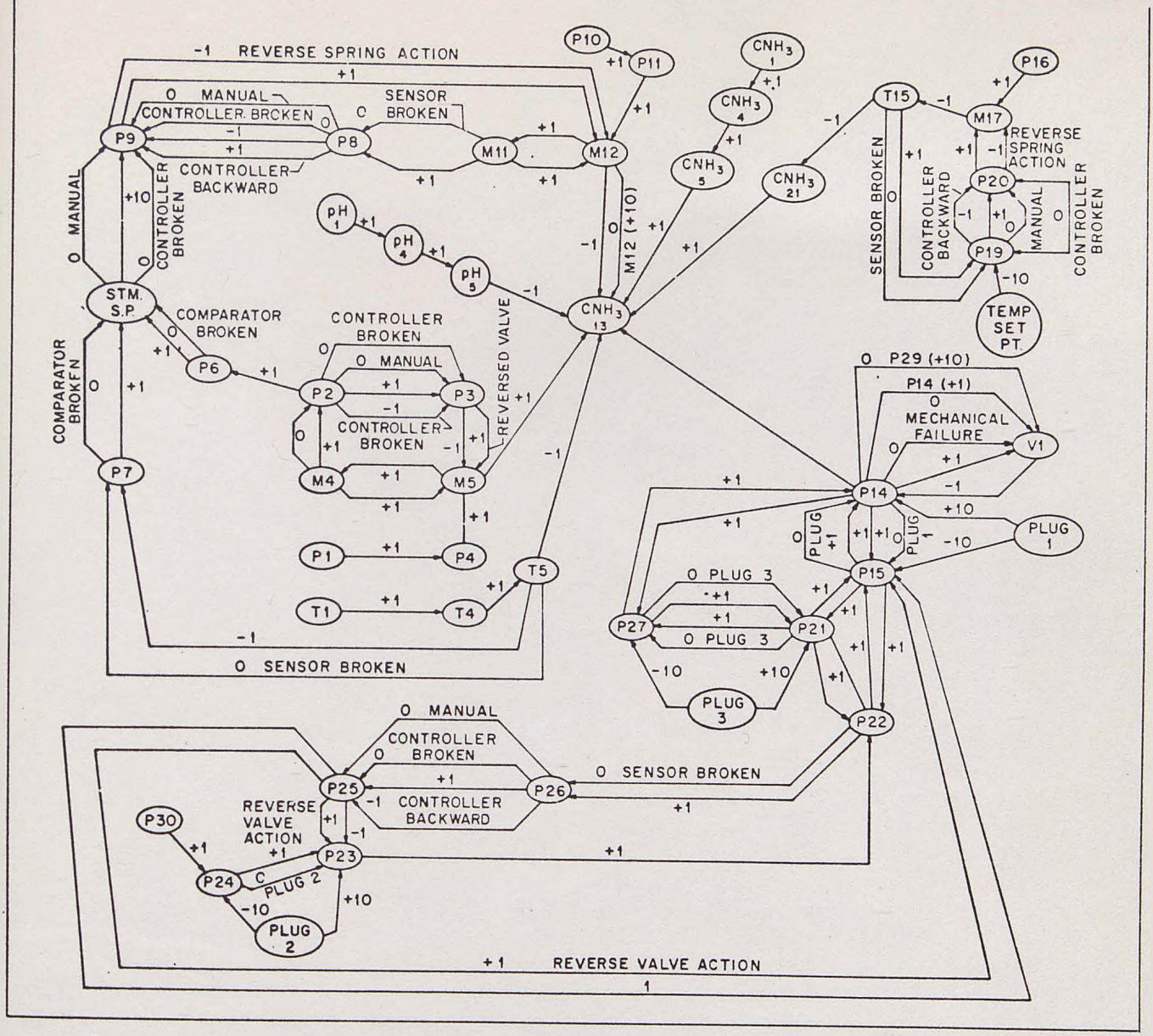


Figure 4. Digraph for sour water stripping system.

One major problem in this approach is the generation of the sets of precursor events. These sets must contain "all" the important events in the proper logical relationships. The generation of fault trees can be a very time consuming and error-prone procedure. For example, in an average chemical process there will commonly be over 50 hazardous events. This large number of incidents is because a single generic hazard, such as release of toxic material, could occur at a number of different locations within the process.

Each fault tree will often require two to three man-days to carry out the generation, documentation, and computation of probabilities of failure. Hence, several man-years of effort are commonly required to carry out a fault tree analysis. A recent study by the U.S. Atomic Energy Commission (WASH-1400) required over 25 man-years to generate the fault trees for one boiling water reactor and one pressurized water reactor. In addition, high quality people are required to carry out the analysis. The magnitude of the time required for analysis has retarded the growth of the fault tree method in the chemical process industry. We are currently developing procedures and computer programs (FTS) that we hope will greatly reduce the time required for quantitative safety analysis.

Symbolic process simulation

The FTS program is essentially a symbolic process simulation. Following identification and evaluation of process hazards, a symbolic model of the complete process is assembled from models of individual pieces of equipment within the process. Models that have been developed by experts represent the latest thinking on normal and failed behavior of processing equipment. The models are signed digraphs. We have developed an algorithm that deduces the fault tree directly from the properties of the digraph. Once the fault tree has been generated, it is placed in minimal cut-set form and the probability of the top event is computed.

The flow diagram is entered in a manner similar to other flow diagram simulators. The equipment and streams are numbered and the topology of the process network defined. The characteristics of each piece of equipment and each stream are also entered. The program constructs a multilinked data structure that contains the information on the process.

Potential hazards are identified by considering the physical and chemical properties of the species within the process and the strength of the equipment. The species

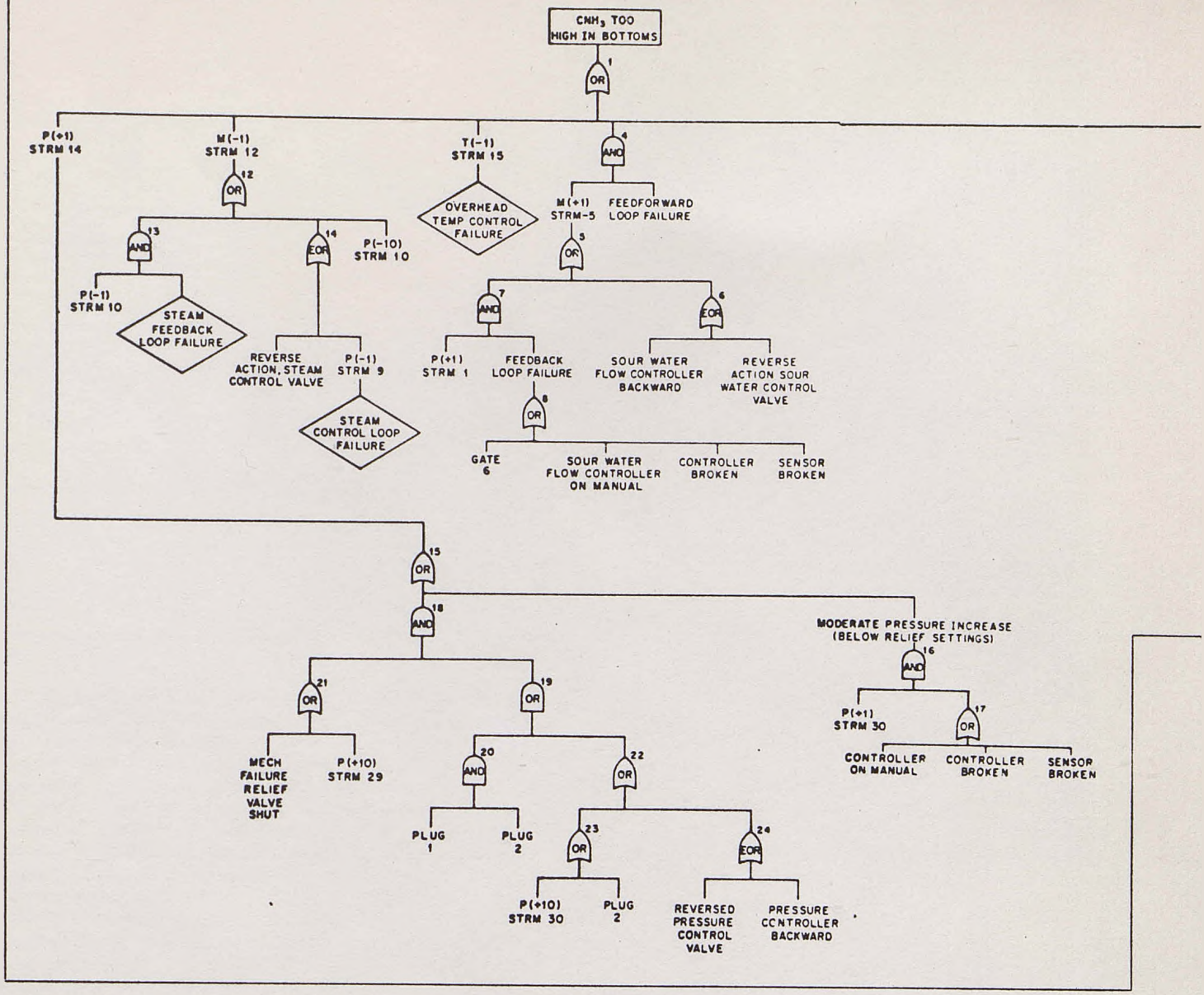


Figure 5. Fault tree for the event "concentration of NH_3 too high" for a sour water stripping system.

hazards are determined from 1) single species property data such as detonation tendencies, toxicity, flammability, explosivivity, etc. and 2) binary species data.

Binary data are required to determine whether species within the process react with each other and if so under what conditions. A binary species reaction matrix is constructed from data in the program's library and from the program user. Equipment related hazards are identified by considering the pressure, temperature, and corrosion ratings of the equipment. The result of hazard identification is a list of hazards and the conditions required for their occurrence. Figure 2 gives the hazard "equation" for explosion.

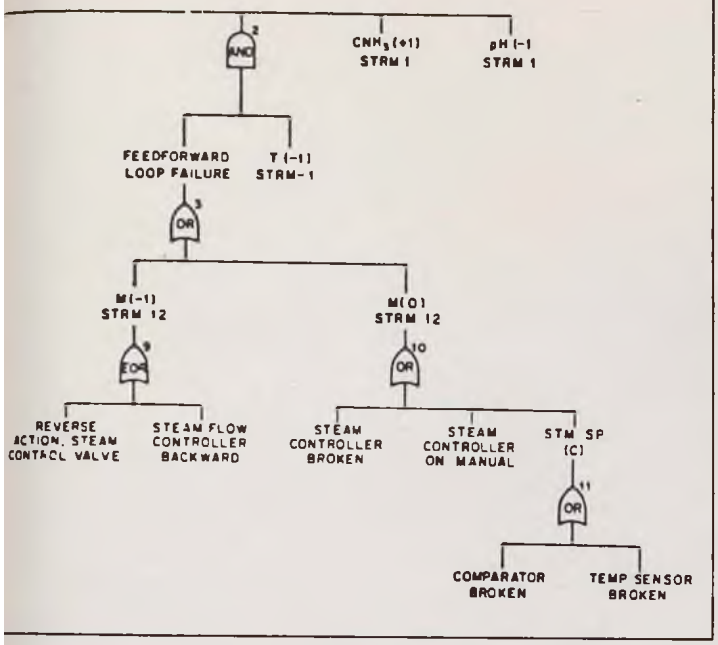
How costs are determined

The total cost of each potential hazard is determined in several ways. First, a detailed calculation of the blast wave radius and high radiation fire radius are computed. Equipment, people, and raw materials within these radii are used in the cost evaluation. Business interruption expenses are computed from product rates and values, and the current delivery times for major pieces of process equipment. The impact of the release of toxic material is based upon the magnitude of the release, plume spread calculations, and toxic nature of the chemical species. These features of the cost of a hazard are then combined to measure the total potential cost for each hazard. These estimates are used as a first approximation of the cost of a hazard. More detailed estimates of each hazard will commonly follow the generation of the fault trees.

The program also contains several rating schemes used by major insurance and chemical companies in the U.S. These schemes use a relative rating scale for species, equipment, etc. The ratings are based on past loss data and intuition. These methods give a very rapid means for screening potential total process hazards, but they are usually not discriminating enough to use on specific equipment faults within a process.

Following the evaluation of hazards within the process, a ranked list of hazards is constructed. What remains is to determine the probability of each hazard.

Signed digraphs are used as models for faults in the FTS program. Models have been prepared for pieces of equipment normally found in chemical processes (reactors, pumps, pipes, mixers, tees, valves, sensors, controllers,



etc.). Each module is an input/output matrix for the piece of equipment. The inputs are variables whose information could flow into the equipment.

In addition to the normal inputs, all known failure modes are considered as inputs. That is, if a failure occurs, how does it change an output variable? Finally, any changes in relationships between input and output variables due to failure modes are indicated. For example, plugging of a pipe changes the input/output relation for pressure in that pipe.

Given a particular hazard equation and process flow diagram, the FTS program assembles a digraph for the complete process. The assembly of the process digraph requires consideration of the gains and time constants for each input/output relationship and for loops within the digraph. The digraph is analyzed to determine the dominant loops with respect to gains and dynamics.

This reduced digraph is converted to a signed digraph where the gains are $+1, 0, -1, +10$, and -10. The gains of ± 10 indicate large changes which exceed the capacity of corrective actions such as occur with negative feedback loops. Modules also have been developed for the human operator's actions in the sensing of variables, computation of control action, and taking control action. External failure models that predict the propagation of failures outside of the process pipes are also being developed.

Synthesizing the fault tree

Lapp has developed an algorithm which deduces the correct system fault tree from a signed digraph for the process. The algorithm is based on a general classification of cut sets in signed digraphs. The key features of the algorithm follow:

1) The topology of the digraph is extremely important. Negative feedback and feedforward loops are detected and their elements determined. Cases of nested loops are also considered. 2) Conditional expansion of events is performed. That is, certain events may preclude others from occurring. The test for these conditions is performed at each expansion of an event. 3) The changes in relationships between variables due to failures are included. 4) Common cause failures are detected directly from the digraph. 5) Human operator actions are included. 6) Large deviations from normal conditions that alter relationships between variables are considered. 7) Events which have been previously developed are detected and copied. This results in a large computation time savings.

Fault tree synthesis by this algorithm is rapid. Trees containing 100 gates are generated in 10 cpu sec. on an IBM-360-67 computer. Following generation, the trees are "drawn" on a line printer.

The fault tree is passed to a subroutine which first places the tree in its minimal cut set form. An algorithm similar to that used by Fussell and Vesely (*3*) has been developed for this task. Once in cut set form, probability calculations are performed. At present, simple deterministic probability values are used. We are extending the algorithm to handle time dependent failure rates with repair.

Consider the flow diagram given in Figure 3. Ammonia and hydrogen sulfide are being steam stripped from a refinery stream. The bottoms from the column go to a biological waste treatment facility. The "bugs" used in this facility are very sensitive to the concentration of ammonia in the waste. Several "kills" of the bugs have occurred due to misoperation oı failures of the stripper system. These losses of the waste disposal unit have caused fairly large releases of organics to the river into which the unit discharges.

The stripper system contains several control loops for maintaining the constant operation of the unit. A reduced digraph for the concentration of ammonia in the bottoms, Figure 4, was constructed from unit models for the valves, sensors, controllers, stripper column, pressure relief valve, etc., that make up the stripper system. The fault tree for this digraph is given in Figure 5.

From the fault tree and probability data on the causing events it is possible to determine the important sets of events which might lead to high ammonia concentrations. Corrective designs could then be considered if the probability of occurrence proved to be too high. #

Literature cited

1. Powers, G. J., and F. C. Tompkins, Jr., "A Synthesis Strategy for Fault Trees in Chemical Processing Systems," Loss Prevention, **8**, *CEP* Technical Manual, AIChE, New York (1974).
2. Fussell, J. B., G. J. Powers, and R. G. Bennetts, "Fault Trees—A State of the Art Discussion," *IEEE Trans. on Reliability*, **R-23** (1) (April, 1974).
3. Fussell, J. B., and W. E. Vesely, "A New Methodology for Obtaining Cut Sets for Fault Trees," *Trans. Am. Nuclear Soc.*, **15** (1) (1972).

G. J. Powers is associate professor of chemical engineering and director, Design Research Center, at Carnegie-Mellon Univ. He has worked for the Ethyl Corp., Dow Chemical Co., and taught for several years at M.I.T. A consultant to numerous chemical companies in the area of process safety analysis, he has published several papers on the safety analysis of chemical processes and is the co-author of the text, "Process Synthesis." Powers earned his B.S.Ch.E. from the Univ. of Michigan and his Ph.D. from the Univ. of Wisconsin.

S. Lapp is research associate, Dept. of Chemical Engineering, Carnegie-Mellon Univ. Currently involved in the development of computer programs to aid in the generation and analysis of fault trees for chemical processes, Lapp has published several papers on fault tree analysis. He earned his B.S.Ch.E. at Carnegie-Mellon Univ.

APPENDIX C

ADDITIONAL COMMENTS

Completeness of the Algorithm

The fault tree synthesis algorithm presented in this paper has been successfully applied to over one hundred systems. Although no formal proof has been presented for the technique, the author believes it to be complete based on the results of the tests run on it.

Program Description

A listing of FTS follows this section. The function of the main program and subroutines are presented below.

MAIN PROGRAM

1. Reads digraph input and top event definition.
2. Controls selection of appropriate subroutines for synthesis of the tree.
3. Calls for output when tree is complete.
4. A number of statements used for debugging purposes have been left in the program. These allow one to examine intermediate results of the program's execution.
5. All subroutines are called directly by the main program.

SUBROUTINE PATH1

1. Creates the path matrix for each connecting node and stores it in IPTH.

SUBROUTINE CYCLE1

1. Finds all negative feedback and feedforward loops through the variable currently under development and stores them in IC.

SUBROUTINE ORGT1

1. Determines the appropriate values for inputs to the variable currently being developed.
2. Creates an OR gate and then stores the input variables and their values in it.
3. None of the inputs are stored for later consistency checks.

SUBROUTINE ORGT2

1. Sets us the first level of gates for the negative feedback or feedforward loop(s) operator.
2. If this is the first entry on the negative feedback loop, the variable is stored for consistency checks. If a negative feedforward loop is involved, the starting variable on it is stored for consistency checks.

SUBROUTINE ORGT3

1. Sets us an OR gate and stores any local zero edge conditions on negative feedback or feedforward loops in it. The next variable back on the loop is also stored with a zero value.

SUBROUTINE ORGT4

1. Creates an OR gate and stores any local zero edges on negative feedback or feedforward loops in it. The next variable back on the loop is also stored with a zero value.
3. This subroutine is only called if the variable currently being developed has a zero value.

SUBROUTINE ORGT5

1. Creates an EOR gate and stores any local inversion edge conditions in it. The next variable back on the negative feedback or feedforward loop is stored with its computed value.

SUBROUTINE ANGT1

1. This subroutine creates an AND gate. The placement of this gate and its inputs are determined by ORGT2 and the MAIN PROGRAM depending on whether negative feedback or feedforward loops are involved.

SUBROUTINE CLNUP

1. This subroutine removes any single feed or no feed gates from the tree and renumbers accordingly.

SUBROUTINE OUTPT

1. This subroutine prints out the fault tree.

```
LINE? SOURCE.FTS1
0000100        DIMENSION IPTH(500,100),IPTF(500),ICSF(50),IC(30,20),-
0000200       1ICF(30),IV(40)
0000300        COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000400       1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000500        COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000600       1IGTVL(500,18)
0000700        READ(5,101) IP,IF
0000800 101    FORMAT(2I3)
0000900        DO 10 I=1,IP
0001000        READ(5,102) IFD(I)
0001100        J=IFD(I)
0001200        READ(5,103) (IPROP(I,K),K=1,J)
0001300 10     READ(5,103) (IGAIN(I,K),K=1,J)
0001400 102    FORMAT(I2)
0001500 103    FORMAT(10I5)
0001600        DO 20 I=1,IF
0001700 20     READ(5,104) IFAIL(I)
0001800 104    FORMAT(I5)
0001900        READ(5,105) IG,IGV
0002000 105    FORMAT(2I5)
0002100        IP1=1000+IP
0002200        ICT=0
0002300        IGF=0
0002400        IGMX=0
0002500        IGMN=1
0002600        CALL PATH(IP,IFD,IPROP,IPTH,IPTF)
0002700 700    N=IG-1000
0002800        CALL CYCLE(IP,IFD,IPROP,IPTF,IPTH,IC,ICN,ICF,N)
0002900        IF(ICN .EQ.0) GO TO 100
0003000        DO 30 I=1,ICN
0003100        IGN=1
0003200        IC1=N
0003300        JF=ICF(I)
0003400        DO 40 J=1,JF
0003500        KF=IFD(IC1)
0003700        DO 50 IC2=1,KF
0003800        IF(IPROP(IC1,IC2) .EQ. IC(I,J)) GO TO 200
0003900 50     CONTINUE
0004000 200    IGN=IGN/IGAIN(IC1,IC2)
0004100 40     IC1=IC(I,J)-1000
0004200        JFF=IFD(IC1)
0004300        DO 60 IC2=1,JFF
0004400        IF(IPROP(IC1,IC2) .EQ. IG) GO TO 300
0004500 60     CONTINUE
0004600 300    IGN=IGN/IGAIN(IC1,IC2)
0004700        IF(IGN .LT. 0) GO TO 400
0004800 30     CONTINUE
0004900 100    IOPT=0
0005000        CALL ORGT1(IOPT,II,J)
```

```
0005010 C       DEBUG
0005020         WRITE(6,901)
0005030 901     FORMAT(' ORGT1')
0005040         CALL OUTPT
0005050 C       DEBUG
0005100 500     IGF=IGF+1
0005200         IF(IGF .GT. IGATE(IGMN,3)) GO TO 600
0005300         IG1=IGF+3
0005400         IG2=IGF+1
0005500         IG=IGATE(IGMN,IG1)
0005600         IGV=IGTVL(IGMN,IG2)
0005700         IF(IG .GT. IP1) GO TO 500
0005800         IF(IG .LT. 1001) GO TO 500
0005900         DO 70 I=1,IGMX
0006000         IF(IG .NE. IGATE(I,1)) GO TO 70
0006100         IF(IGV .NE. IGTVL(I,1)) GO TO 70
0006200         ITMP=I
0006210         IMIN=0
0006220         IMAX=0
0006230 620     KG=IGATE(ITMP,3)+3
0006240         IF(KG .LT. 4) GO TO 320
0006250         DO 51 K=4,KG
0006260         IF(IGATE(ITMP,K) .GT. 0) GO TO 51
0006270         IF(IMIN .LT. 1) GO TO 720
0006280         DO 61 M=1,IMIN
0006290         IF(IV(M) .EQ. -IGATE(ITMP,K)) GO TO 51
0006300 61      CONTINUE
0006310 720     IMAX=IMAX+1
0006320         IV(IMAX)=-IGATE(ITMP,K)
0006330         IF(IGMN .EQ. IV(IMAX)) GO TO 420
0006340 51      CONTINUE
0006350         IMIN=IMIN+1
0006360 320     IF(IMIN .GT. IMAX) GO TO 520
0006370         ITMP=IV(IMIN)
0006380         GO TO 620
0006381 420     IF(IGF .EQ. IGATE(IGMN,3)) GO TO 820
0006382         IG2=IGATE(IGMN,3)+2
0006383         DO 71 K=IG1,IG2
0006384         IGATE(IGMN,K)=IGATE(IGMN,K+1)
0006385 71      IGTVL(IGMN,K-2)=IGTVL(IGMN,K-1)
0006386 820     IGATE(IGMN,3)=IGATE(IGMN,3)-1
0006387         IGF=IGF-1
0006388         GO TO 500
0006389 520     IGATE(IGMN,IG1)=-I
0006390         IGTVL(IGMN,IG2)=0
0006391         GO TO 500
0006500 70      CONTINUE
0006600         GO TO 700
0006700 600     IGMN=IGMN+1
0006800         IF(IGMN .GT. IGMX) GO TO 610
0006900         IGF=0
0007000         GO TO 500
0007100 610     CALL CLNUP
0007120 C         DEBUG
```

```
0007140         ENDFILE 6
0007160 C         DEBUG
0007180         IF(IGMX .EQ. 0)GO TO 220
0007200         CALL OUTPT
0007300         STOP
0007320 220     WRITE(6,106)
0007340 106     FORMAT(15HONO GATES EXIST)
0007360         STOP
0007400 400     IF(ICT .EQ. 0) GO TO 800
0007500         DO 80 II=1,ICT
0007600         J9=JF+1
0007700         JJF=ICSF(II)
0007800         IF(J9 .NE. JJF) GO TO 80
0007900         DO 90 J=1,JJF
0008000         IF(ICST(II,J) .EQ. IG) GO TO 900
0008100 90      CONTINUE
0008200         GO TO 80
0008300 900     DO 11 K=1,JF
0008400         JN=J+K
0008500         IF(JN .GT. JJF) JN=JN-JJF
0008600         IF(ICST(II,JN) .NE. IC(I,K)) GO TO 80
0008700 11      CONTINUE
0008800         GO TO 210
0008900 80      CONTINUE
0009000 800     ICT=ICT+1
0009100         ICSF(ICT)=JF
0009200         ICST(ICT,1)=IG
0009300         IC1=N
0009400         ICSVAL(ICT,1)=IGV
0009600         DO 21 J=1,JF
0009700         JJ=J+1
0009800         ICST(ICT,JJ)=IC(I,J)
0010000         KF=IFD(IC1)
0010100         DO 31 K=1,KF
0010200         IF(IPROP(IC1,K) .EQ. IC(I,J)) GO TO 310
0010300 31      CONTINUE
0010400 310     ICSVAL(ICT,JJ)=ICSVAL(ICT,J)/IGAIN(IC1,K)
0010500 21      IC1=IC(I,J)-1000
0010600         ICSF(ICT)=JJ
0010700         II=ICT
0010800         J=1
0010900 810     CALL ORGT2(II,J,IPH)
0010910 C       DEBUG
0010920         WRITE(6,902)
0010930 902       FORMAT(' ORGT2')
0010940          CALL OUTPT
0010950 C        DEBUG
0011000         CALL ANGT1
0011100         IOPT=1
0011110 C        DEBUG
0011120          WRITE(6,903)
0011130 903       FORMAT(' ANGT1')
0011140           CALL OUTPT
0011150 C         DEBUG
```

```
0011200        CALL ORGT1(IOPT,II,J)
0011210 C       DEBUG
0011220         WRITE(6,901)
0011230          CALL OUTPT
0011240 C      DEBUG
0011300        CALL ORGT3(II,J,IPH)
0011310 C       DEBUG
0011320         WRITE(6,904)
0011330 904     FORMAT(' ORGT3')
0011340          CALL OUTPT
0011350 C       DEBUG
0011400        GO TO 500
0011500 210    IF(IGV .NE. 0) GO TO 410
0011600        IF(ICSF(II) .NE. J) GO TO 510
0011700 710    CALL ORGT4(II)
0011710 C       DEBUG
0011720         WRITE(6,905)
0011730 905     FORMAT(' ORGT4')
0011740          CALL OUTPT
0011750 C       DEBUG
0011800        GO TO 500
0011900 510    CALL ORGT5(II,J)
0011910 C       DEBUG
0011920         WRITE(6,906)
0011930 906     FORMAT(' ORGT5')
0011940         CALL OUTPT
0011950 C        DEBUG
0012000        GO TO 500
0012100 410    IF(ICSF(II) .EQ. J) GO TO 710
0012200        JF=IFD(N)
0012300        I5=ICST(II,J+1)
0012400        DO 41 M1=1,JF
0012500        IF(IPROP(N,M1) .EQ. I5) GO TO 41
0012600        IF(IPROP(N,M1) .GT. 1000) GO TO 810
0012700        IF(IPROP(N,M1) .GT. IF) GO TO 910
0012800        ITP1=IPROP(N,M1)
0012850        ITP=IFAIL(ITP1)
0012900        IF(ITP-I5) 810,41,810
0013000 910    IG6=IGV/IGAIN(N,M1)
0013100        IF(IG6 .EQ. 1) GO TO 810
0013200 41     CONTINUE
0013300        GO TO 510
0013400        END
LINE? SOURCE.PATH1
0000100        SUBROUTINE PATH(IP,IFD,IPROP,IPTH,IPTF)
0000200        DIMENSION IPROP(500,10),IPTH(500,100),IFD(500),IPTF(50
0000300        DO 10 I=1,IP
0000400        IMIN=1
0000500        IMAX=IFD(I)
0000700        DO 20 J=1,IMAX
0000800 20     IPTH(I,J)=IPROP(I,J)
0000900 100    JT=IPTH(I,IMIN)-1000
0001000        IF(JT .LT. 1) GO TO 200
0001100        IF(JT .GT. IP) GO TO 200
```

```
0001600          JFD=IFD(JT)
0001700          DO 30 J=1,JFD
0002100          DO 40 K=1,IMAX
0002200          IF(IPROP(JT,J) .EQ. IPTH(I,K)) GO TO 30
0002300 40       CONTINUE
0002400          IMAX=IMAX+1
0002500          IPTH(I,IMAX)=IPROP(JT,J)
0002600 30       CONTINUE
0002700 200      IMIN=IMIN+1
0002800          IF(IMIN .GT. IMAX) GO TO 10
0002900          GO TO 100
0003000 10       IPTF(I)=IMAX
0003100          RETURN
0003200          END
LINE? SOURCE.CYCLE1
0000100          SUBROUTINE CYCLE(IP,IFD,IPROP,IPTF,IPTH,IC,ICN,ICF,N)
0000200          DIMENSION IFD(500),IPROP(500,10),IPTH(500,100),IPTF(50
0000300         1IC(30,20),ICF(30)
0000400          ICN=0
0000500          IM=IPTF(N)
0000550          IX=1000+N
0000600          DO 10 I=1,IM
0000700          IF(IPTH(N,I) .EQ. IX) GO TO 100
0000800 10       CONTINUE
0000900          RETURN
0001000 100      DO 30 I=1,30
0001100 30       ICF(I)=0
0001200          IMIN=1
0001300          IMAX=0
0001400          INF=IFD(N)
0001500          DO 20 I=1,INF
0001520          IF(IPROP(N,I) .LT. 1001) GO TO 20
0001600          JT=IPROP(N,I)-1000
0001620          IF(JT .GT. IP) GO TO 20
0001700          JF=IPTF(JT)
0001800          DO 40 J=1,JF
0001900          IF(IPTH(JT,J) .NE. IX) GO TO 40
0002000          IMAX=IMAX+1
0002100          ICF(IMAX)=1
0002200          IC(IMAX,1)=IPROP(N,I)
0002300          GO TO 20
0002400 40       CONTINUE
0002500 20       CONTINUE
0002600 400      INN=ICF(IMIN)
0002700          IN=IC(IMIN,INN)
0002720          INN=INN-1
0002730          IF(INN .EQ. 0) GO TO 300
0002740          DO 80 I=1,INN
0002760          IF(IN .EQ. IC(IMIN,I)) GO TO 500
0002780 80       CONTINUE
0002800          IF(IN .NE. IX) GO TO 300
0002850          ICN=ICN+1
0002900          IMIN=IMIN+1
0003000          IF(IMIN .GT. IMAX) GO TO 600
```

```
0003100          GO TO 400
0003105 500      IF(IMIN .EQ. IMAX) GO TO 600
0003110          MM=IMIN
0003115          M=IMIN+1
0003120          DO 5 I=M,IMAX
0003125          ICF(MM)=ICF(I)
0003130          INN=ICF(MM)
0003135          DO 90 J=1,INN
0003140 90       IC(MM,J)=IC(I,J)
0003145 5        MM=MM+1
0003150          IMAX=IMAX-1
0003155          GO TO 400
0003200 300      L=0
0003250          IN=IN-1000
0003300          INF=IFD(IN)
0003400          DO 50 I=1,INF
0003450          IF(IPROP(IN,I) .LT. 1001) GO TO 50
0003500          JT=IPROP(IN,I)-1000
0003600          IF(JT .GT. IP) GO TO 50
0003700          JF=IPTF(JT)
0003800          DO 60 J=1,JF
0003900          IF(IPTH(JT,J) .NE. IX) GO TO 60
0004000          IF(L .GT. 0) GO TO 200
0004100          ICF(IMIN)=ICF(IMIN)+1
0004200          ICT=ICF(IMIN)
0004300          IC(IMIN,ICT)=IPROP(IN,I)
0004400          L=1
0004500          GO TO 50
0004600 200      ICT=ICF(IMIN)-1
0004700          IMAX=IMAX+1
0004800          DO 70 K=1,ICT
0004900 70       IC(IMAX,K)=IC(IMIN,K)
0005000          ICT=ICT+1
0005100          ICF(IMAX)=ICT
0005200          IC(IMAX,ICT)=IPROP(IN,I)
0005300          GO TO 50
0005400 60       CONTINUE
0005500 50       CONTINUE
0005600          GO TO 400
0005620 600      IF(ICN .EQ. 0) RETURN
0005640          DO 11 I=1,ICN
0005660 11       ICF(I)=ICF(I)-1
0005680          RETURN
0005700          END
LINE? SOURCE.XORGT1
0000100          SUBROUTINE ORGT1(IOPT,II,J)
0000200          COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000300         1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000400          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000500         1IGTVL(500,18)
0000600          IGMX=IGMX+1
0000700          IPRV=0
0000800          IF(IOPT .EQ. 0) GO TO 200
0000900          IPRV=ICST(II,J+1)
```

```
0001000          IGATE(IGMX,1)=0
0001100          IGTVL(IGMX,1)=0
0001200          GO TO 300
0001300 200      IGATE(IGMX,1)=IG
0001400          IGTVL(IGMX,1)=IGV
0001500 300      IGATE(IGMX,2)=1
0001600          M2=IFD(N)
0001700          L=3
0001800          DO 10 I=1,M2
0001900          IF(IPROP(N,I) .EQ. IPRV) GO TO 10
0002000          IF(IPROP(N,I) .GT. 1000) GO TO 100
0002100          IF(IPROP(N,I) .LE. IF) GO TO 10
0002200          IF(IGAIN(N,I) .NE. IGV) GO TO 10
0002300 100      L=L+1
0002400          IGATE(IGMX,L)=IPROP(N,I)
0002500          LL=L-2
0002600          IGTVL(IGMX,LL)=IGV/IGAIN(N,I)
0002700 10       CONTINUE
0002800          IGATE(IGMX,3)=L-3
0002850          IF(ICPT .GT. 0) RETURN
0002900          IF(IGF .EQ. 0) RETURN
0003000          IG1=IGF+3
0003100          IG2=IGF+1
0003200          IGATE(IGMN,IG1)=-IGMX
0003300          IGTVL(IGMN,IG2)=0
0003400          RETURN
0003500          END
LINE? SOURCE.XORGT2
0000100          SUBROUTINE ORGT2(II,J,IPH)
0000200          COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000300         1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000400          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000500         1IGTVL(500,18)
0000550          ICHK=0
0000600          IGMX=IGMX+1
0000700          IMT=IGMX
0000800          IGATE(IGMX,1)=IG
0000900          IGATE(IGMX,2)=1
0001000          IGTVL(IGMX,1)=IGV
0001100          IGATE(IGMX,3)=2
0001200          JF=IFD(N)
0001300          IPRV=ICST(II,J+1)
0001350          IPH=IPRV
0001400          DO 10 I1=1,JF
0001500          IF(IPROP(N,I1) .GT. IF) GO TO 10
0001600          ITP=IPROP(N,I1)
0001700          IF(IFAIL(ITP) .NE. IPRV) GO TO 10
0001800          IF(IGAIN(N,I1) .EQ. 0) GO TO 10
0001900          IF(ICHK .GT. 0) GO TO 300
0001930          ICHK=1
0001960          IGATE(IGMX,4)=-(IGMX+1)
0001990          IGATE(IGMX,5)=-(IGMX+3)
0002020          IGTVL(IGMX,2)=0
0002050          IGTVL(IGMX,3)=0
```

```
0002200          IGMX=IGMX+1
0002250          IPH=-IGMX
0002300          IGATE(IGMX,1)=0
0002400          IGTVL(IGMX,1)=0
0002500          IGATE(IGMX,2)=2
0002600          IGATE(IGMX,3)=2
0002700          IGATE(IGMX,4)=IPRV
0002800          IGATE(IGMX,5)=-(IGMX+1)
0002900          IGTVL(IGMX,3)=0
0003000          IGTVL(IGMX,2)=ICSVAL(II,J+1)
0003010          IGMX=IGMX+1
0003020          IGATE(IGMX,1)=0
0003030          IGTVL(IGMX,1)=0
0003040          IGATE(IGMX,2)=1
0003050          IGATE(IGMX,3)=0
0003060 300      IGATE(IGMX,3)=IGATE(IGMX,3)+1
0003070          L=IGATE(IGMX,3)+3
0003080          IGATE(IGMX,L)=ITP
0003090          IGTVL(IGMX,L-2)=1
0003200 10       CONTINUE
0003250          IF(ICHK .GT. 0) GO TO 200
0003300          IGATE(IGMX,4)=IPRV
0003400          IGTVL(IGMX,2)=ICSVAL(II,J+1)
0003500          IGATE(IGMX,5)=-(IGMX+1)
0003600          IGTVL(IGMX,3)=0
0003700 200      IF(IGF .EQ. 0) RETURN
0003800          IG1=IGF+3
0003900          IG2=IGF+1
0004000          IGATE(IGMN,IG1)=-IMT
0004100          IGTVL(IGMN,IG2)=0
0004200          RETURN
0004300          END
LINE? SOURCE.XORGT3
0000100          SUBROUTINE ORGT3(II,J,IPH)
0000200          COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000300         1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000400          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000500         1IGTVL(500,18)
0000600          IGMX=IGMX+1
0000700          IGATE(IGMX,1)=0
0000800          IGTVL(IGMX,1)=0
0000900          IGATE(IGMX,2)=1
0000950          IGATE(IGMX,3)=2
0001000          L=1
0001100          IGATE(IGMX,4)=IPH
0001200          IGATE(IGMX,5)=-(IGMX+1)
0001300          JF=IFD(N)
0001400          IGTVL(IGMX,2)=ICSVAL(II,J+1)
0001450          IF(IPH .LT. 0) IGTVL(IGMX,2)=0
0001500          IGTVL(IGMX,3)=0
0001510          IGMX=IGMX+1
0001520          IGATE(IGMX,1)=0
0001530          IGTVL(IGMX,1)=0
0001540          IGATE(IGMX,2)=1
```

```
0001550          IGATE(IGMX,4)=ICST(II,J+1)
0001560          IGTVL(IGMX,2)=0
0001600          DO 10 I=1,JF
0001700          IF(IPROP(N,I) .GT. IF) GO TO 10
0001800          ITP=IPROP(N,I)
0001900          IF(IFAIL(ITP) .NE. IGATE(IGMX,4)) GO TO 10
0001950          IF(IGAIN(N,I) .NE. 0) GO TO 10
0002000          L=L+1
0002100          IGATE(IGMX,L+3)=ITP
0002200          IGTVL(IGMX,L+1)=1
0002300 10       CONTINUE
0002400          IGATE(IGMX,3)=L
0002500          RETURN
0002600          END
LINE? SOURCE.XORGT4
0000100          SUBROUTINE ORGT4(II)
0000200          DIMENSION ITS(10)
0000300          COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000400         1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000500          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000600         1IGTVL(500,18)
0000700          JF=IFD(N)
0000800          L=0
0000850          ITP2=ICST(II,1)
0000900          DO 10 I=1,JF
0001000          IF(IPROP(N,I) .GT. IF) GO TO 10
0001100          ITP=IPROP(N,I)
0001200          IF(IFAIL(ITP) .NE. ITP2) GO TO 10
0001300          IF(IGV .EQ. 0) GO TO 100
0001400          IF(IGAIN(N,I) .EQ. 0) GO TO 10
0001500          IGN=IGV/IGAIN(N,I)
0001600          IF(IGN .NE. ICSVAL(II,1)) GO TO 10
0001700          L=L+1
0001800          ITS(L)=ITP
0001900          GO TO 10
0002000 100      IF(IGAIN(N,I) .NE. 0) GO TO 10
0002100          L=L+1
0002200          ITS(L)=ITP
0002300 10       CONTINUE
0002400          IF(L .NE. 0) GO TO 200
0002500          IF(IGF .EQ. IGATE(IGMN,3)) GO TO 400
0002600          IG1=IGF+4
0002700          IG2=IGATE(IGMN,3)+3
0002800          DO 30 I=IG1,IG2
0002900          IJ=I-1
0003000          IGATE(IGMN,IJ)=IGATE(IGMN,I)
0003100 30       IGTVL(IGMN,IJ-2)=IGTVL(IGMN,I-2)
0003200 400      IGATE(IGMN,3)=IGATE(IGMN,3)-1
0003300          IGF=IGF-1
0003350          RETURN
0003400 200      IF(L .NE. 1) GO TO 300
0003500          IGATE(IGMN,IGF+3)=ITS(1)
0003600          IGTVL(IGMN,IGF+1)=1
0003700          RETURN
```

```
0003800 300    IGMX=IGMX+1
C003900        IGATE(IGMX,1)=0
C004000        IGTVL(IGMX,1)=0
0004100        IGATE(IGMX,2)=1
0004200        IGATE(IGMX,3)=L
0004300        DO 20 I=1,L
0004400        IGATE(IGMX,I+3)=ITS(I)
C004500 20     IGTVL(IGMX,I+1)=1
C004600        RETURN
0004700        END
LINE? SOURCE.XORGT5
C000100        SUBROUTINE ORGT5(II,J)
0000200        COMMON /AREA1/ IPROP(500,10),IGAIN(500,10),IFD(500),-
0000300       1N,ICST(50,20),ICSVAL(50,20),IF,IFAIL(100)
0000400        COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000500       1IGTVL(500,18)
0000600        IGMX=IGMX+1
0000650        IMT=IGMX
0000700        IGATE(IGMX,1)=IG
C000800        IGTVL(IGMX,1)=IGV
C000850        IGATE(IGMX,3)=2
C000900        IGATE(IGMX,2)=2
C000950        IF(IGV .EQ. 0) IGATE(IGMX,2)=1
0001000        IGATE(IGMX,4)=ICST(II,J+1)
C001050        IGATE(IGMX,5)=-(IGMX+1)
0001100        JF=IFD(N)
0001200        DO 10 I1=1,JF
0001300        IF(IPROP(N,I1) .EQ. IGATE(IGMX,4)) GO TO 100
C001400 10     CONTINUE
0001500 100    IGTVL(IGMX,2)=IGV/IGAIN(N,I1)
0001550        IGTVL(IGMX,3)=0
0001560        IGMX=IGMX+1
0001570        IGATE(IGMX,1)=0
C001580        IGTVL(IGMX,1)=0
0001590        IGATE(IGMX,2)=1
0001600        L=3
0001700        DO 20 I1=1,JF
C001800        IF(IPROP(N,I1) .GT. IF) GO TO 20
C001900        ITP=IPROP(N,I1)
0002000        IF(IFAIL(ITP) .NE. IGATE(IGMX-1,4)) GO TO 20
0002100        IF(IGV .EQ. 0) GO TO 200
C002200        IF(IGAIN(N,I1) .EQ. 0) GO TO 20
0002300        IGN=IGV/IGAIN(N,I1)
0002400        IF(IGN .EQ. ICSVAL(II,J+1)) GO TO 20
0002500        GO TO 300
0002600 200    IF(IGAIN(N,I1) .NE. 0) GO TO 20
0002700 300    L=L+1
C002800        IGATE(IGMX,L)=ITP
C002900        IGTVL(IGMX,L-2)=1
0003000 20     CONTINUE
0003100        IGATE(IGMX,3)=L-3
0003120        IG1=IGF+3
0003140        IG2=IGF+1
0003160        IGATE(IGMN,IG1)=-IMT
```

```
0003180          IGTVL(IGMN,IG2)=0
0003200          RETURN
0003300          END
LINE? SOURCE.XANGT1
0000100          SUBROUTINE ANGT1
0000200          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000300         1IGTVL(500,18)
0000400          IGMX=IGMX+1
0000500          DO 10 I=1,3
0000600  10      IGTVL(IGMX,I)=0
0000700          IGATE(IGMX,1)=1
0000800          IGATE(IGMX,2)=3
0000900          IGATE(IGMX,3)=2
0001000          DO 20 I=4,5
0001100  20      IGATE(IGMX,I)=-(IGMX+I-3)
0001200          RETURN
0001300          END
LINE? SOURCE.XCLNUP
0000100          SUBROUTINE CLNUP
0000200          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000300         1IGTVL(500,18)
0000350          IF(IGMX .EQ. 0) RETURN
0000400  600     IGM=IGMX
0000500          DO 10 I=1,IGM
0000550          IZZ=IGATE(I,3)
0000600          IF(IZZ-1) 100,200,10
0000700  200     ITS1=IGATE(I,4)
0000800          ITS2=IGTVL(I,2)
0000900  100     IF(I .EQ. IGM) GO TO 300
0001000          J=I+1
0001100          DO 20 JJ=J,IGM
0001200          J1=JJ-1
0001300          DO 30 I1=1,3
0001400  30      IGATE(J1,I1)=IGATE(JJ,I1)
0001500          IGTVL(J1,1)=IGTVL(JJ,1)
0001600          JF=IGATE(JJ,3)+3
0001700          IF(JF .EQ. 3) GO TO 20
0001800          DO 40 I1=4,JF
0001900          IGATE(J1,I1)=IGATE(JJ,I1)
0002000  40      IGTVL(J1,I1-2)=IGTVL(JJ,I1-2)
0002100  20      CONTINUE
0002200  300     IGMX=IGMX-1
0002250          IF(IGMX .EQ. 0) RETURN
0002300          I2=-I
0002310          IPR1=1
0002320          IPR2=IGMX
0002330          IPCL=0
0002400          IF(IZZ .GT. 0) GO TO 800
0002500  510     DO 50 J=IPR1,IPR2
0002600  500     JF=IGATE(J,3)+3
0002700          IF(JF .EQ. 3) GO TO 50
0002800          DO 60 J1=4,JF
0002900          IF(IGATE(J,J1) .NE. I2) GO TO 60
0003000          IF(J1 .EQ. JF) GO TO 400
```

```
0003100          J2=J1+1
0003200          DO 70 J3=J2,JF
0003300          J4=J3-1
0003400          IGATE(J,J4)=IGATE(J,J3)
0003500 70       IGTVL(J,J4-2)=IGTVL(J,J3-2)
0003600 400      IGATE(J,3)=IGATE(J,3)-1
0003700          GO TO 500
0003800 60       CONTINUE
0003900 50       CONTINUE
0004000          IF(IPCL-1) 900,800,800
0004050 800      IPR3=IPR1
0004150          DO 80 J=IPR3,IGMX
0004200          JF=IGATE(J,3)+3
0004300          IF(JF .EQ. 3) GO TO 80
0004400          DO 90 J1=4,JF
0004500          IF(IGATE(J,J1) .NE. I2) GO TO 90
0004510          DO 71 M=4,JF
0004520          IF(IGATE(J,M) .NE. ITS1) GO TO 71
0004530          IF(IGTVL(J,M-2) .NE. ITS2) GO TO 71
0004540          IPCL=2
0004550          IPR1=J
0004560          IPR2=J
0004570          GO TO 510
0004580 71       CONTINUE
0004600          IGATE(J,J1)=ITS1
0004700          IGTVL(J,J1-2)=ITS2
0004800 90       CONTINUE
0004900 80       CONTINUE
0004910 900      DO 11 KK=1,IGMX
0004920          JF=IGATE(KK,3)+3
0004930          IF(JF .EQ. 3) GO TO 11
0004940          DO 21 LL=4,JF
0004950          IF(IGATE(KK,LL) .GT. I2) GO TO 21
0004960          IGATE(KK,LL)=IGATE(KK,LL)+1
0004970 21       CONTINUE
0004980 11       CONTINUE
0005000          GO TO 600
0005100 10       CONTINUE
0005200          RETURN
0005300          END
LINE? SOURCE.XOUTPT
0000100          SUBROUTINE OUTPT
0000200          COMMON /AREA2/ IG,IGV,IGMX,IGF,IGMN,IGATE(500,20),-
0000300         1IGTVL(500,18)
0000400          DO 10 I=1,IGMX
0000500          MM=IGATE(I,3)+3
0000600          WRITE(6,201) I,(IGATE(I,J),J=1,MM)
0000700          NN=MM-2
0000800          WRITE(6,202) (IGTVL(I,K),K=1,NN)
0000900 10       CONTINUE
0001000 201      FORMAT(///13HOGATE NUMBER ,I3//,1X,20I5)
0001100 202      FORMAT(1HO,20I5)
0001200          RETURN
0001300          END
```