Miller, J. A.; Bowman, C. T. Mechanism and Modeling of Nitrogen Chemistry in Combustion. *Prog. Energy Combust. Sci.* **1989,** *15* (4), 287–338.
Miller, J. A.; Bowman, C. T. Kinetic Modeling of the Reduction of Nitric Oxide in Combustion Products by Isocyanic Acid. *Int. J. Chem. Kinet.* **1991,** *23,* 289.
Myerson, A. L. The Reduction of Nitric Oxide in Simulated Combustion Effluents by Hydrocarbon-Oxygen Mixtures. *Symp. (Int.) Combust.,* [*Proc.*] **1974,** *15,* 1085.
Myerson, A. L.; Taylor, F. R.; Faunce, B. G. Ignition Limits and Products of the Multistage Flames of Propane-Nitrogen Dioxide Mixtures. *Symp. (Int.) Combust.,* [*Proc.*] **1957,** *6,* 154.
Rimpel, G.; Just, Th. Unpublished work cited in Roth and Just (1984), 1980.
Roth, P.; Just, Th. Kinetics of the High Temperature, Low Concentration CH_4 Oxidation Verified by H and O Atom Measurements. *Symp. (Int.) Combust.,* [*Proc.*] **1984,** *20,* 807.
Seeker, Vm. R. Design of Advanced Reburning Systems. *Proceedings of the Reburning Workshop,* Örenäs Slott, Sweden, Nov 26–27, 1990; Nordic Gas Technology Centre: Hørsholm, Denmark, 1990.
Tsang, W.; Hampson, R. F. Chemical Kinetic Data Base for Combustion Chemistry. Part I. Methane and Related Compounds. *J. Phys. Chem. Ref. Data* **1986,** *15,* 1087.
Tully, F. P.; Perry, R. A.; Thorne, L. R.; Allensdorf, M. D. Free-Radical Oxidation of Isocyanic Acid. *Symp. (Int.) Combust.,* [*Proc.*] **1988,** *22,* 1101.
Warnatz, J. Rate Coefficients in the C/H/O System. In *Combustion Chemistry;* Gardiner, W. C., Jr., Ed.; Springer-Verlag: New York, 1984; Chapter 5, pp 197–346.
Wendt, J. O. L.; Sterling, C. V.; Matovich, M. A. Reduction of Sulfur Trioxide ad Nitrogen Oxides by Secondary Fuel Injection. *Symp. (Int.) Combust.,* [*Proc.*] **1973,** *14,* 897.
Westley, F.; Herron, J. T.; Cvetanovic, R. J.; Hampson, R. F.; Mallard, W. C. *NIST Chemical Kinetics Database,* Version 3.0; National Institute of Standards and Technology: Gaithersburg, MD, April 1991.

Received for review August 20, 1991
Accepted December 17, 1991

New Developments of the Digraph-Based Techniques for Fault-Tree Synthesis

Chuei-Tin Chang* and Her-Chuan Hwang
Department of Chemical Engineering, National Cheng Kung University, Tainan, Taiwan 70101, Republic of China

By making use of steady-state analysis of the fault propagation behaviors in process systems, improvements of the digraph-based techniques for fault-tree synthesis have been achieved. In particular, generalized sub-tree structures have been developed for variables associated with nodes on feedforward and feedback loops and a preliminary loop-searching computer algorithm has been worked out. These techniques are integrated in a fault-tree construction procedure which is also presented in this paper. This revised procedure has been implemented in a number of application examples. The results show that the proposed approach is not only feasible but also capable of identifying correct causes of the top event which are missing in the fault trees obtained by the conventional operators.

Introduction

Increasing pressure from the society, more and more stringent government regulations, and the trend toward large and complex chemical plants have led to an unprecedented demand for a rigorous approach for identifying the causes and likelihood of the occurrences of undesirable events within a process. One of the principal methods used for hazard identification and assessment is fault-tree analysis (FTA) (Lee et al., 1985). Although FTA has been successfully applied in the chemical process industries, its use has not become as widespread as its earlier proponents had predicted. One of the fundamental problems appears to be that the synthesis of fault trees is a tedious and difficult task which requires skills beyond those normally possessed by process engineers. Thus, it is attractive to develop a systematic, computer-based method for fault-tree construction as an aid.

To facilitate the synthesis of fault trees, a variety of strategies have been devised; e.g. those employed transfer functions (Fussell, 1973), decision tables (Salem et al., 1977; Kumamoto and Henley, 1979), digraphs (Lapp and Powers, 1977), reliability graphs (Camarda et al., 1978), signal-flow graphs (Kumamoto et al., 1981) and mini-fault trees (Kelly and Lees, 1986a–d; Khan and Hunt, 1989), etc. In essence, the approaches adopted in these studies are quite similar. These suggested fault-tree construction algorithms are all developed from qualitative system failure models of one form or another, which can be assembled by connecting the failure models of the individual components (units) in the system. One of the most popular models used by the practitioners of FTA is the digraph. Numerous studies concerning the applications and modifications of this technique have been published in the literature (Shaeiwitz et al., 1977; Chamow, 1978; Lambert, 1979; Lapp and Powers, 1979; Allen and Rao, 1980; Cummings et al., 1983; Allen, 1984; Andrews and Morgan, 1986; Andrews and Brennan, 1990). A digraph provides an intermediate step which gives explicit relationships between the process variables, human errors, and equipment failure events, from which the fault tree can be constructed. This method offers a structured approach which is well suited for modeling systems with control loops, where a set of sub-tree structures (operators) can be applied.

In general, the digraph-based techniques are quite powerful and have been applied to a number of realistic systems. However, there are some pitfalls which have been encountered by others while using this approach (Andow, 1980, 1981; Galluzzo and Andow, 1984). Although some of these problems have been addressed recently (Shafaghi et al., 1984; Powers and Lapp, 1987), difficulties still exist in the application of this method to complex process systems. This is mainly caused by the tangled control and *process* feedforward (FFL) and feedback (FBL) loops in these systems. Incorrect fault trees may be generated if the existing techniques are applied carelessly. On the other

* To whom all correspondence should be addressed.

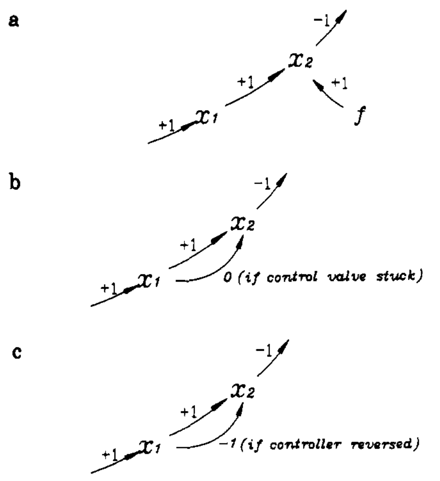


Figure 1. (a) Digraph representation of a type A fault or failure. (b) Digraph representation of a type C failure. (c) Digraph representation of a type D failure.

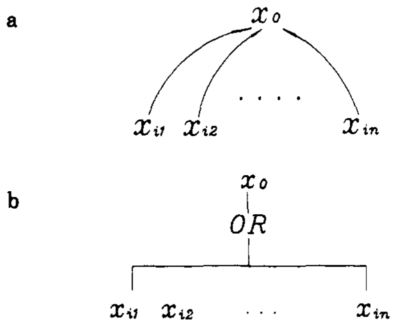


Figure 2. (a) Digraph representing the cause–effect relations of fault propagation. (b) The fault tree structure (structure I) corresponding to digraphs with the configuration of a tree.

hand, the signed directed graph (SDG) has long been utilized as a tool for developing fault diagnosis techniques (Iri et al., 1979; Shiozaki et al., 1985). In recent years, additional new insights in using SDG to model fault propagation behaviors in process systems have been published (Kramer and Palowitch, 1987; Oyeleye and Kramer, 1988; Chang and Yu, 1990; Qian, 1990). Thus, there are incentives for a detailed study to explore the possibility of revamping the conventional digraph-based fault-tree synthesis algorithm at the present time.

On the basis of a steady-state analysis of the fault propagation behaviors in chemical process plants, the digraph-based techniques have been improved in this study. More specifically, generalized fault-tree structures (operators) corresponding to various digraph configurations, i.e. tree, feedforward loop, and feedback loop, have been developed for systems with both control and process loops. Furthermore, since the failure models for a realistic system are so complicated that manual identification and classification of loops becomes unreliable, a preliminary computer algorithm has been completed in this work to overcome this difficulty. Finally, the Lapp–Powers algorithm for fault-tree synthesis has been modified and has been applied to several realistic examples to demonstrate the feasibility and effectiveness of the suggested approach.

Classification of Faults and Failures

One of the primary objectives of constructing a fault tree is to identify the mechanisms by which the undesirable top event could occur. Thus, a detailed steady-state analysis of the effects of the initiating basic events, i.e. the faults and failures, becomes extremely helpful for this purpose. As mentioned before, a digraph explicitly describes the qualitative cause–effect relationships between the deviations in process variables (represented by 0, ±1, or ±10) and component failures (represented by 0, 1, or 10) (Lapp and Powers, 1977). In order to qualitatively simulate the propagation of faults/failures in process systems using digraphs, one has to be able to appropriately model these initiating events and their effects. Generally speaking, the faults and failures in a process plant can be classified into four types based on their digraph representations and, also, the patterns of their propagation in the system, as follows.

Type A. For faults such as disturbances in the process variables or partial component failures (i.e. degradation in the equipment's performance) such as a small leak or a partial plug in a control valve, the corresponding digraph representation should be a node without inputs. The outward edges of such nodes are directed to process variables. A typical digraph model can be found in Figure 1a, where x_1 and x_2 are process variables and f is the fault or failure of type A. The effects of these types of faults/failures can be determined by assigning a nonzero value (±1 or ±10) to f, and the values of the other variables in the digraph can then be evaluated accordingly. Notice that, in an analysis of these effects for the purpose of classification, the implied assumption is that no other failures exist simultaneously. Thus, the value of x_2 may be affected not only by f but also by x_1, *only* if both x_1 and x_2 are on the same FBL.

Type B. The digraph configuration of component failures such as sensor failing high or control valve failing close is actually the same as that of type A. However, their effects should be analyzed differently. If a failure of type B (f) occurs, then x_2 is always affected by f only, even when both x_1 and x_2 are on the same FBL.

Type C. Component failures such as sensor stuck or control valve stuck should be modeled by conditional edges with zero gain. An example can be found in Figure 1b. The occurrence of a failure of this type only changes the configuration of the system digraph; i.e. the edge between x_1 and x_2 can be considered as nonexistent. The state variables of the system remain at the normal levels without additional disturbances.

Type D. Component failures such as controller reversed (from direct action to reverse action or vice versa) or control value reversed (from air to open to air to close or vice versa) can also be represented by conditional edges. An example of such failures is presented in Figure 1c, which is also represented by a change in the configuration only. Obviously, the occurrence of a failure of type D changes the direction of the effects of an additional fault (if it occurs) propagating from x_1 to x_2.

Implications of Loops in Fault-Tree Synthesis

The problem of representing fault propagation mechanisms in a fault tree is now considered. In the process of fault-tree synthesis, the input and output events represented by a digraph are, in general, connected by the logic relation of an OR gate. For example, if one considers the digraph model presented in Figure 2a, then the corresponding fault tree can be obtained accordingly (Figure 2b). In other words, the effect of the occurrence of one input event is considered to be independent of that of another. The realization of any of the input events alone can cause the output event. However, if the output is a node on a loop in the digraph model, structure I is no longer applicable.

From a purely structural view point, two types of "loops" are important for constructing fault trees, i.e. feedforward

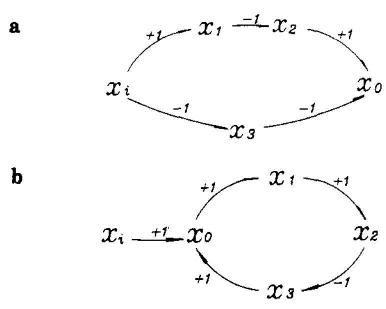


Figure 3. (a) Negative feedforward loop (NFFL). (b) Negative feedback loop (NFBL).

loops (FFL—two or more paths from one node in a digraph to another different node in the digraph) and feedback loops FBL—a path through the nodes in a digraph which starts and terminates at the same node). The first case considered is when an output event x_o is the terminal node of a "negative" feedforward loop (NFFL); i.e. the signs of the products of the edge gains on the paths in a FFL are not the same. A typical example is presented in Figure 3a. Then, any change in the variable x_i which starts the FFL creates two opposite effects simultaneously. Without further information about the system, one cannot determine the sign of deviation (+, −, or 0) of x_o based on the digraph model alone.

Next, consider the case when x_o is on a negative feedback loop (NFBL), i.e. a FBL on which the product of the edge gains around the loop is negative (see Figure 3b). Notice that, although the gain of the edge between x_i and x_o is positive, the product of the gains on the path $x_i \rightarrow x_o \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_o$ is negative. Again, two opposite effects on x_o are generated by any change in x_i. Thus, erroneous fault trees will be produced if structure I, which includes only the logic relation of an OR gate, is employed for systems with NFFL and/or NFBL.

Remember that a digraph is nothing but a qualitative model of the physical system which represents the relationships among the process variables. Loops are formed as a result of assembling various different component digraphs in the system. These loops can also be classified according to their functions, i.e. control loops and process loops. The control loops in a digraph can be identified easily by searching for the nodes that represent the variables in control systems. Since these systems are specified by process and instrumentation engineers during the design stage of a chemical plant to achieve certain operational requirements, their locations in a digraph should be considered as known information. On the other hand, the process loops are caused by the complex relations among system variables. These loop variables can be operating parameters (e.g. pressure and temperature) in the process and, also, signals in control systems (e.g. air pressure to a control valve). In general, one cannot identify these process loops directly from a piping and instrumentation diagram (P&ID).

Therefore, in applying the digraph-based techniques to synthesize fault trees for design a realistic process system, one is often confronted with the problems of handling feedforward and feedback and the task of identifying and classifying various different loop configurations in a digraph.

Generalized Fault-Tree Structure for Feed-Forward Loops

The digraph in Figure 4a is considered next, which represents the behavior of a fictitious system. Notice that this digraph is a process feedforward loop. As mentioned before, the emphasis of earlier studies is mainly on control loops. There is no formal procedure established for process feedforward loops or FFLs in general. Thus, without other available tools, one is forced to use structure I in this case. The result is presented in Figure 4b. One can see that the basic event "$x_i(+1)$" appears in both branches of the fault tree. Despite the fact that the same events are repeated, the minimum cut set is acceptable, i.e. $\{x_i(+1)\}$. However, if one applies structure I to the negative process feedforward loop presented in Figure 5a, the result in Figure 5b can be obtained. Clearly, since the minimum cut sets $\{x_i(+1)\}$ and $\{x_i(-1)\}$ are mutually exclusive, this fault tree is erroneous.

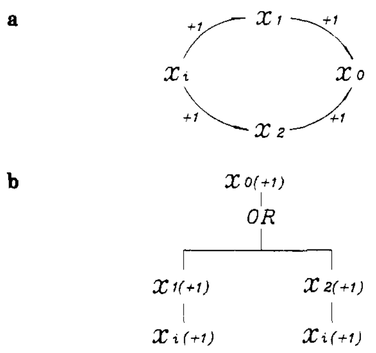


Figure 4. (a) Positive feedforward loop (PFFL). (b) Corresponding fault tree obtained by structure I.

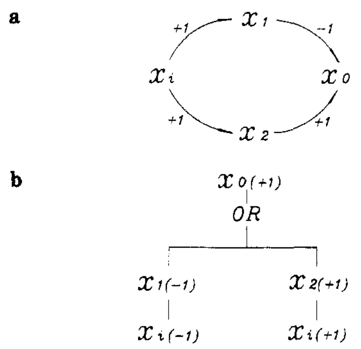


Figure 5. (a) Negative feedforward loop. (b) Corresponding fault tree obtained by structure I.

From the definition of a FFL, it is quite obvious that the disturbances that enter the starting node of a FFL can propagate through various different paths to affect the variable associated with the terminal node. In a (negative) control feedforward loop, these effects will cancel out as a result of the regulatory action. A well-established "operator" has been used to develop fault trees in this situation (Powers and Lapp, 1987). However, without additional process information, the net effect of this disturbance is indeterminable in a negative *process* feedforward loop. Further, this net effect may be changed if the values of some of the gains on the FFL vary because of events such as equipment failure and/or human error.

In this research, component malfunctions that reverse the signs of edge gains (a type D failure) on the paths of a FFL were not considered, because these kinds of failures can be almost always eliminated by preventive inspection before the startup of the system. Also, since the probability of simultaneous occurrences of two or more component failures of type B, C, or D in a FFL is quite low, the possibility of such events was ignored in constructing the corresponding fault trees. On the basis of the above assumptions and the operator previously used for control loops (Powers and Lapp, 1987), a modified version of the existing technique has been developed in this study for

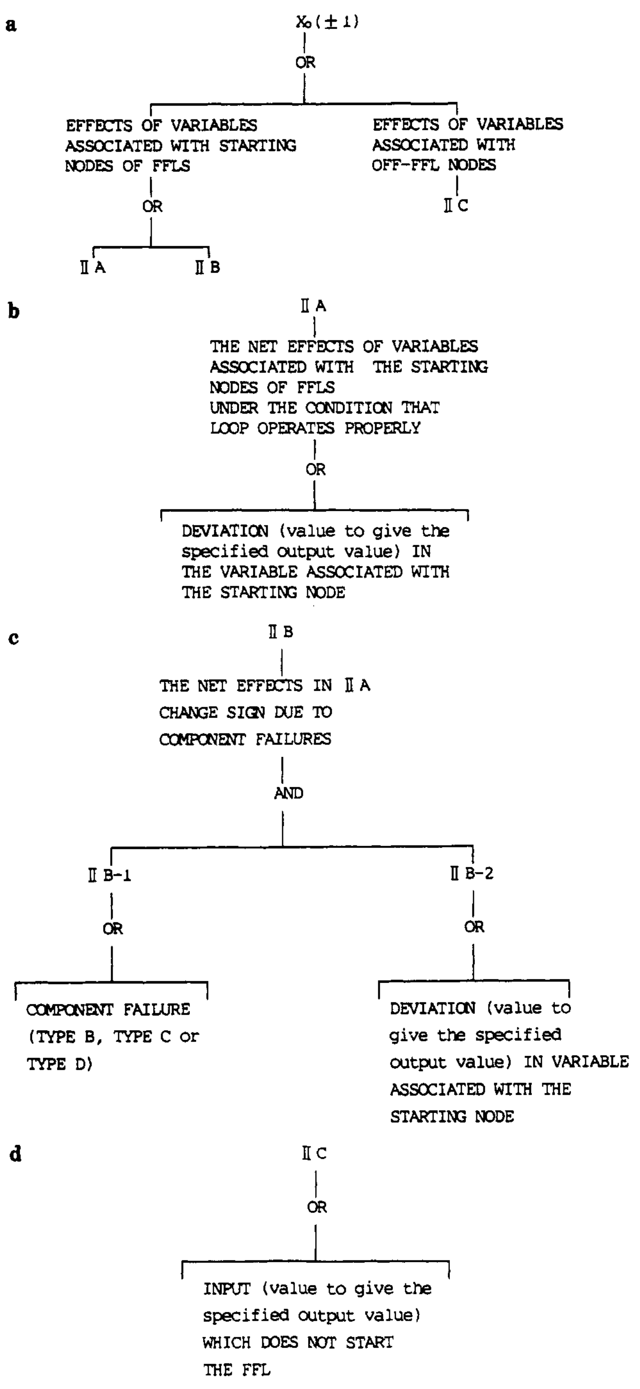

a
X₀(±1)
OR
EFFECTS OF VARIABLES ASSOCIATED WITH STARTING NODES OF FFLS
EFFECTS OF VARIABLES ASSOCIATED WITH OFF-FFL NODES
OR
II C
II A
II B
b
II A
THE NET EFFECTS OF VARIABLES ASSOCIATED WITH THE STARTING NODES OF FFLS UNDER THE CONDITION THAT LOOP OPERATES PROPERLY
OR
DEVIATION (value to give the specified output value) IN THE VARIABLE ASSOCIATED WITH THE STARTING NODE
c
II B
THE NET EFFECTS IN II A CHANGE SIGN DUE TO COMPONENT FAILURES
AND
II B-1
II B-2
OR
OR
COMPONENT FAILURE (TYPE B, TYPE C or TYPE D)
DEVIATION (value to give the specified output value) IN VARIABLE ASSOCIATED WITH THE STARTING NODE
d
II C
OR
INPUT (value to give the specified output value) WHICH DOES NOT START THE FFL

Figure 6. (a) Generalized fault tree structure (structure II) used for the output variable deviation (±1) associated with the terminal node of a FFL. (b) substructure IIA. (c) Substructure IIB. (d) Substructure IIC.

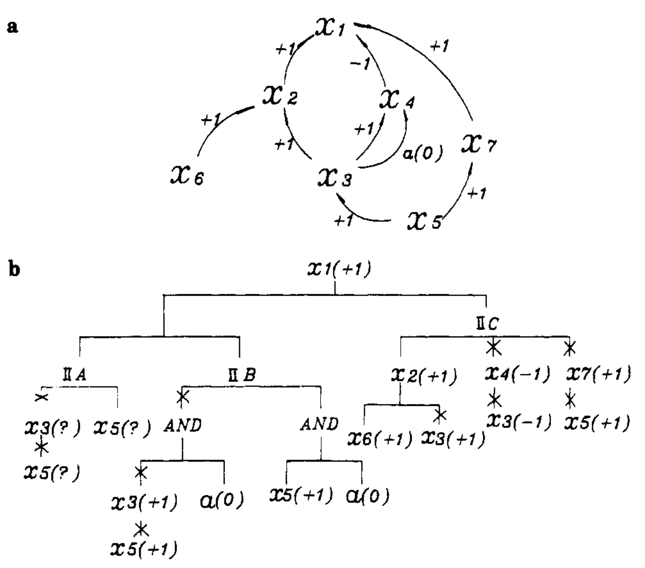


Figure 7. (a) A complex FFL. (b) Corresponding fault tree obtained by structure II.

variables associated with the terminal nodes of FFLs in general (see Figure 6a–d).

This generalized fault-tree structure is applicable to both control and process FFLs. In the case of control FFLs, since the net effects of deviations "±1" in the starting node can be regarded as zero, the events that appear in substructure IIA should be those corresponding to deviations "±10". As a result, fault trees obtained by structure II are the same as those constructed by the conventional operator (Andrews and Morgan, 1986; Powers and Lapp, 1987). On the other hand, since deviations ±1 in the variable associated with the starting node of a process FFL may generate a net effect of magnitude 1, fault trees of a different form can be produced by applying the same structure II. In the implementation of this structure for synthesizing fault trees of positive process FFLs, the drawback of structure I, i.e. repeated events, can be avoided and, thus, the resulting trees can be simplified greatly. When it is applied to negative process FFLs, the erroneous logic relations created by structure I can also be eliminated completely. Further, it can be shown that when applied to complex FFLs with three or more paths, this proposed approach yields more appropriate fault trees than those obtained by the conventional operator.

In synthesizing fault trees by structure II, one may find that the inputs to substructures IIA and IIB are not basic events. Naturally, the tree needs to be developed further from these nonbasic events using structure I. Notice that event "$x_0(\pm1)$" or "$x_0(\pm10)$" in structure II may be associated with the terminal node of more than one FFLs in digraph. Thus, it is possible that, in the process of developing the nonbasic events of substructures IIA and IIB, another starting node is encountered. If this is the case, the corresponding input event should be removed from the fault tree. In addition, if its output does not have any other input attached, the output should be deleted also. The same steps are repeated until the above condition, i.e. the output is without inputs after deletion, cannot apply. Similarly, in the process of developing the nonbasic events of substructure IIC, the same procedure should be carried out if a starting node of another FFL is reached.

The fictitious system in Figure 7a is used as an example. Here, "$a(0)$" represents a type C failure. The corresponding fault tree can be easily developed using structure II (see Figure 7b). A more detailed flow chart of the fault-tree synthesis procedure can also be found in Figure 14 of this paper. Notice that the symbols "×" indicate the branches in the fault tree which must be removed as a result of implementing the deletion procedure mentioned in the previous paragraph. The symbol "?" denotes that the sign of deviation associated with the starting node of a negative FFL is indeterminable. Generally speaking, the net effect of FFLs can be estimated from a mathematical model of the system, on-line measurement data, or operating experiences of the plant engineers and operators. Notice that negative process FFLs of the same structure may represent systems with the same P&ID but the net effects may be opposite in direction due to different design parameters and/or operating conditions (Hwang, 1991). Thus, the values of ? may be interpreted as "+", "−", or "0" and should be assessed on a case-by-case basis using

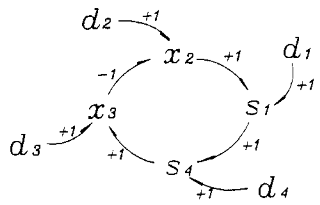


Figure 8. Digraph configuration of a typical NFBL.

Table I. New Interpretations of Gains between Regular Loop Variables

gain	input	output
+1	⊕	⊕
	⊖	⊖
−1	⊕	⊖
	⊖	⊕

additional quantitative process information.

Generalized Fault-Tree Structure for Feedback Loops

First the fault-tree construction approach adopted in this work for positive FBLs is outlined. In the case of process loops, the regular structure I should be applied except a termination criterion must be introduced to avoid repetition in the fault tree. The positive control FBLs are caused mainly by incorrect design or installation mistakes. As mentioned in the previous section, these faults can be detected by proof-testing before startup and, thus, have been neglected in all examples of this study due to their low probability.

Second the fault propagation behaviors in a standard negative control FBL presented in Figure 8 are analyzed. In this figure, s_1 is the sensor signal, x_2 represents the controlled variable, x_3 is the manipulated variable, s_4 denotes the output signal from the controller, and d_1, d_2, d_3, and d_4 represent the disturbances which enter the control loop at their respective locations. As mentioned before, any one of these disturbances generates two opposite effects, and, in this case, the net effect is zero if the control loop functions properly. For example, without loop component failures, the event "$d_2(+1)$" causes "$x_2(0)$", "$s_1(0)$", "$s_4(+1)$", and "$x_3(1+)$" at the new steady state. This special behavior of negative control FBLs creates a problem in simulating fault propagation; i.e. the cause–effect relations are not consistent with the individual edge gains specified in the digraph. Furthermore, if one is interested in identifying all the possible causes of the event $x_3(+1)$, then implementation of the conventional operator fails to serve this purpose. In particular, the fault tree constructed by the existing techniques does not contain some of the apparent reasons for $x_3(+1)$, e.g. $d_2(+1)$ under the condition that the control FBL operates normally.

In this research, the approach suggested by Iri et al. (1977) has been adopted to describe the special behavior of negative FBLs; i.e. two additional states "⊕" and "⊖" are introduced in addition to 0, ±1, and ±10. The symbol ⊕ (or ⊖) is interpreted as the state of a variable which would have a value +1 (or −1) without feedback but does not appear abnormal at the new steady state due to the regulatory action. Using this new set of state values, one can represent the cause–effect relations in a negative FBL more accurately. Notice that although the levels of edge gains remain unchanged, their interpretations must be expanded according to Table I. The original interpretations of gains between the deviations ±1 and ±10 are still allowed and, thus, not repeated in Table I.

Table II. New Interpretations of Gains between Sensor Outputs and Controller Outputs

gain	sensor output	controller output
+1	⊕	+1
	⊖	−1
−1	⊕	−1
	⊖	+1

Table III. Patterns of Variable Deviations in Control FBLs Corresponding to Disturbances of Value +1

disturbance	s_1	x_2	x_3	s_4
$d_1(+1)$	⊕	−1	+1	+1
$d_2(+1)$	⊕	⊕	+1	+1
$d_3(+1)$	⊖	⊖	⊕	−1
$d_4(+1)$	⊖	⊖	⊕	⊕

In addition to the above modifications, the special characteristics of the relation between the sensor output s_1 and the controller output s_4 need to be analyzed further. The cases of proportional (P) and proportional-integral (PI) controllers are considered:

$$s_4(t) = K_c e(t) = K_c s_1(t) \tag{1}$$

$$s_4(t) = K_c\left[e(t) + \frac{1}{\tau_I}\int_0^t e(t)\,dt\right] = K_c\left[s_1(t) + \frac{1}{\tau_I}\int_0^t s_1(t)\,dt\right] \tag{2}$$

In the above equations, the set-point value s_1^{sp} is assumed to be equal to the value of s_1 at normal operation, which is zero by definition. Since $e(t) = s_1(t) - s_1^{sp}$, the error $e(t)$ reduces to $s_1(t)$ in both equations. A well-known property of P-controllers is that offset may exist when a disturbance of constant magnitude enters the system. However, if the controller gain K_c is high enough, this offset can be reduced to a level which should be regarded as ⊕ or ⊖. Since the magnitude of K_c is large, it is highly possible that the deviation in s_4 is still significant and should be interpreted as ±1. If a PI-controller is used, one can conclude from eq 2 that the value of s_4 may still be nonzero even when the sensor output returns to its normal level. Thus, the digraph models of controllers in negative FBLs should be treated differently. In particular, additional interpretations of the edge gain between the sensor output s_1 and the controller output s_4 should be introduced according to Table II. Again, only new interpretations are tabulated. The original interpretations of gains ±1 and the ones listed in Table I are still allowed between s_1 and s_4. For the sake of brevity, they are not repeated here. Notice that since the digraphs are not suitable for describing dynamic behaviors, these tabulated values are the ones established at new steady states.

The node at which a disturbance enters the negative control FBL is now considered. As mentioned before, the variable associated with this node is eventually affected in two opposite directions. As a result of the regulatory action of the control loop, the two effects cancel out and the variable should be assigned with the value of ⊕ or ⊖, depending on the direction of the disturbance. The patterns of deviations in the loop variables of the standard NFBL are summarized in Table III. Each row in this table corresponds to the deviations reached at the new steady state after one of the disturbances, $d_i(+1)$, i = 1, 2, 3, 4, is introduced.

Several interesting features can be observed from Table III; i.e. (1) a path is formed by the loop variables with the values ⊕ and/or ⊖, (2) the starting node of this path is the

Table IV. Patterns of Variable Deviations in Control FBLs Corresponding to Disturbances of Value +10

disturbance	s_1	x_2	x_3	s_4
$d_1(+10)$	+1	-10	+10	+10
$d_2(+10)$	+1	+1	+10	+10
$d_3(+10)$	-1	-1	+1	-10
$d_4(+10)$	-1	-1	+1	+1

node where the disturbance enters, and (3) the terminal node is the one corresponding to the sensor output. For illustration purposes, a specific case is considered where disturbance $d_3(+1)$ enters the loop at x_3. The reason for $s_1(\ominus)$ is obvious, since control FBLs are designed to keep the sensor output (not the controlled variable) at the set-point value. On the basis of the new interpretations listed in Table I, the values of x_2 and x_3 should be $\ominus$ and $\oplus$, respectively. The variable x_3 is affected by both d_3 and s_4. Thus, the value of s_4 should be -1 so that the positive effect created by $d_3(+1)$ can be cancelled and the downstream sensor output can be maintained at the normal level. Notice also that the value of s_1 and s_4 match the new interpretation of the gain +1 presented in the second row of Table II.

It may appear that the last pattern in Table III violates the rules specified in Table II. However, if one takes into account the fact that the disturbance $d_4(+1)$ also enters the loop at s_4, this result is in fact quite reasonable. In this case, the positive effect of disturbance $d_4(+1)$ is actually cancelled by the negative effect generated by $s_1(\ominus)$, as implied by the second row of Table II.

Similar analysis can be carried for disturbances of the magnitude 10. The value 10 in this study is regarded as a "very large" quantity according to its original definition suggested by Lapp and Powers (1977). A summary of the patterns of deviations in the loop variables, which are associated with disturbances of value +10, is presented in Table IV. Notice that these disturbances are considered as "uncontrollable" here. By definition, the effects generated by a uncontrollable disturbance cannot be cancelled in a control NFBL. As a result, after one or more loop variables become saturated, a deviation of magnitude 1 occurs in the sensor output. Also note that if a loop variable becomes saturated, the magnitude of its deviation (from the normal level) is regarded as 10 in this work. However, very large deviations in the loop variables do not always have to be saturated. In such cases, 10 is only a qualitative description about a deviation which is significantly larger than 1. Thus, the range of actual value that can be classified as ±10 for each variable should be determined individually. Finally, it should be pointed out that uncontrollable disturbances may not exist under all circumstances. Thus, the capability of control FBLs in the system digraph should be assessed in advance to establish the deviation patterns similar to Table IV on a case-by-case basis.

Actually, the above discussions only provide a qualitative description of the system behavior after faults or failures of type A are introduced in control NFBLs. Naturally, the effects of other types of faults and failures should also be analyzed. Notice that the digraph representation of a component failure of type B is essentially equivalent to that of simultaneous occurrence of a type C failure and a local disturbance. For the case considered in which the magnitude of the disturbance of type B failure is 1, e.g. sensor failing high. Since the value of the loop variable at which the failure or disturbance enters is fixed at +1 (or -1), the patterns of deviations in loop variables should be the same as those specified in Table IV. If the magnitude of the disturbance or the effect of type B failure is 10 (e.g. control valve failing open), the deviations in the loop variable can be determined simply by making use of the conventional interpretations of the edge gains. For example, the event "control valve failing open" produces the following pattern: $x_2(+10)$, $s_1(+10)$, $s_4(+10)$, and $x_3(+10)$.

Although the occurrence of type D failures will not be considered in all examples of this study, it is still desirable to reserve options for including such events in the generalized fault-tree structure. Thus, a brief analysis of their steady-state effects is also presented here. Since, in this case, the NFBL is converted to a PFBL and becomes unstable, the values of loop variables are all driven to +10 or -10 according to the edge gains on the PFBL. For example, the event "controller reversed" creates two possible outcomes, i.e. (i) $s_1(+10)$, $x_2(+10)$, $x_3(-10)$, $s_4(-10)$ and (ii) $s_1(-10)$, $x_2(-10)$, $x_3(+10)$, $s_4(+10)$. Patterns of fault propagation corresponding to other type D failures can be determined in a similar fashion.

For a summary of the analysis presented in this section, a modified fault-tree structure (structure III) has been developed for NFBLs. Its substructures are presented in Figure 9a–c. In substructure IIIA, the part labeled by "(a)" represents component failures that cause reverse gains in NFBLs (type D). As mentioned before, these types of faults are neglected in all examples of this paper. Also, the values of the local disturbances indicated by "(b)" should be determined by using the original standard interpretations of the edge gains.

New features developed in this study are marked by "(c)", "(d)", "(e)", and "(f)". The part indicated by (c) should follow the interpretations provided in Table II; i.e. this branch of the fault tree exists *only when the output x_o is the controller output.* Note that the symbol "$\odot$" represents the value $\oplus$ or $\ominus$. In substructure IIIC, the values of inputs on NFBL (marked (d)) must satsify the cause–effect relations defined in Table I, and, on the basis of Table III, the magnitude of the local disturbance (labeled (e)) to cause the event $x_o(\odot)$ should be 1. Since the loop variables above the controller output in the fault tree are guaranteed not to have the value $\odot$, the patterns in Table III are always consistent with the branches in fault trees developed by structure III.

Notice that when $x_o(+1)$ (or $x_o(-1)$) is used as the top event, results of implementing structure III actually include causes of $x_o(+10)$ also. This is reasonable since, if one is concerned with the hazardous effects of a moderate increase in x_o, $x_o(+10)$ should also be an undesirable event in most cases. However, if mutually exclusive cut sets are identified due to the inclusion of all the causes of both events, the ones corresponding to $x_o(+10)$ should be excluded. Thus, in the process of developing fault trees along NFBLs, special care has to be taken to determine the value of the local disturbance in substructure IIA (marked (f)) after the sensor output is reached. Not only its value should be ±10 (the cause of $x_o(\pm1)$), but also the corresponding pattern of deviations in loop variables (see Table IV) should be consistent with the intermediate events above this disturbance in the fault tree. Thus, from Table III and Table IV, it is clear that part (f) should be omitted when there is a parallel branch in the fault tree created by (c).

Example 1. For illustration of the use of structure III, the heat exchange between a hot stream H and a cold stream C in an exchanger EX_1 is considered. In order to maintain the exit temperature of H at a constant level, the flow rate of H is adjusted via a negative feedback control loop. A simplified flow diagram is presented in Figure 10a.

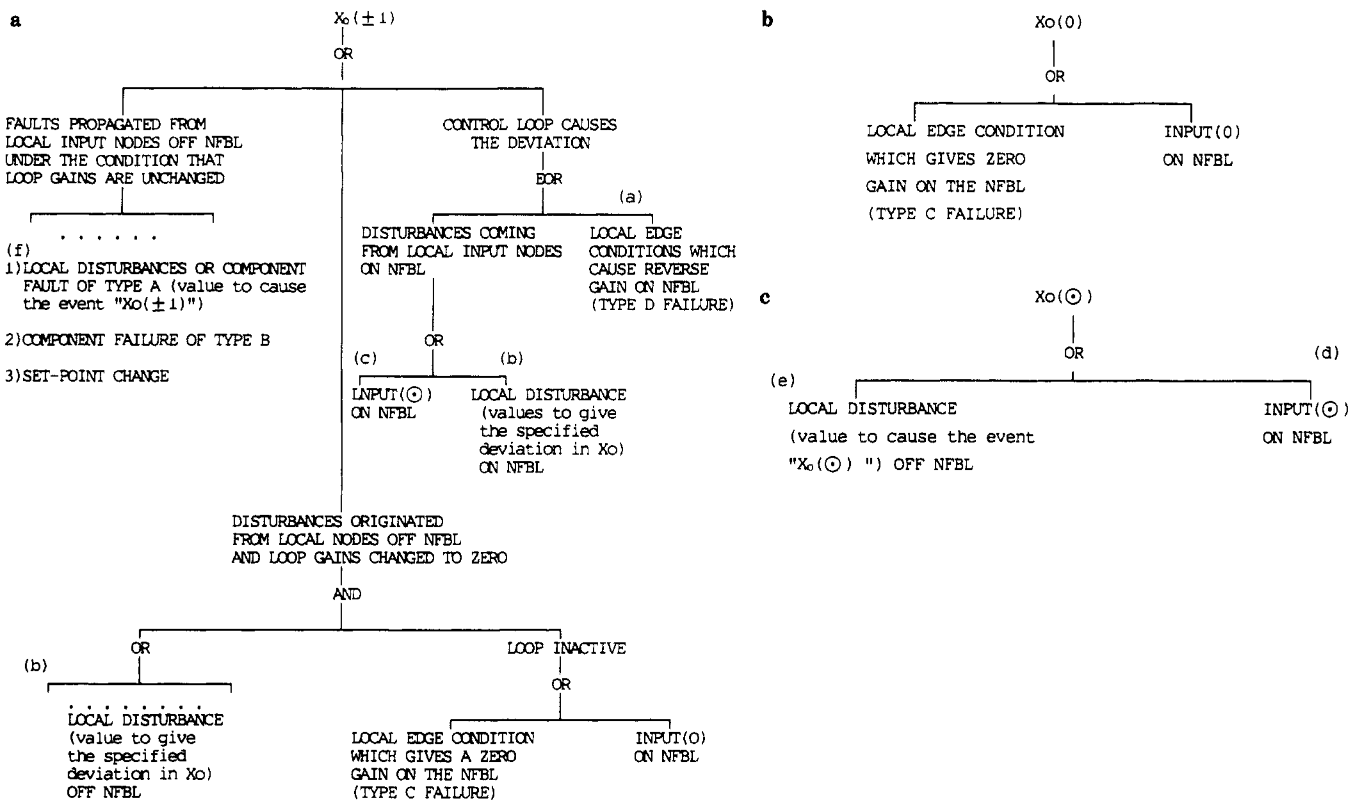

a
Xo(±1)
OR
FAULTS PROPAGATED FROM LOCAL INPUT NODES OFF NFBL UNDER THE CONDITION THAT LOOP GAINS ARE UNCHANGED
(f)
1)LOCAL DISTURBANCES OR COMPONENT FAULT OF TYPE A (value to cause the event "Xo(±1)")
2)COMPONENT FAILURE OF TYPE B
3)SET-POINT CHANGE
CONTROL LOOP CAUSES THE DEVIATION
EOR
(a)
DISTURBANCES COMING FROM LOCAL INPUT NODES ON NFBL
LOCAL EDGE CONDITIONS WHICH CAUSE REVERSE GAIN ON NFBL (TYPE D FAILURE)
OR
(c)
(b)
INPUT(⊙) ON NFBL
LOCAL DISTURBANCE (values to give the specified deviation in Xo) ON NFBL
DISTURBANCES ORIGINATED FROM LOCAL NODES OFF NFBL AND LOOP GAINS CHANGED TO ZERO
AND
OR
(b)
LOCAL DISTURBANCE (value to give the specified deviation in Xo) OFF NFBL
LOOP INACTIVE
OR
LOCAL EDGE CONDITION WHICH GIVES A ZERO GAIN ON THE NFBL (TYPE C FAILURE)
INPUT(O) ON NFBL
b
Xo(0)
OR
LOCAL EDGE CONDITION WHICH GIVES ZERO GAIN ON THE NFBL (TYPE C FAILURE)
INPUT(O) ON NFBL
c
Xo(⊙)
OR
(e)
(d)
LOCAL DISTURBANCE (value to cause the event "Xo(⊙)") OFF NFBL
INPUT(⊙) ON NFBL

Figure 9. (a) Generalized fault tree structure (structure IIIA) used for output variable deviation (±1) associated with a node on NFBL. (b) Generalized fault tree structure (structure IIIB) used for output variable deviation (0) associated with a node on NFBL. (c) Generalized fault tree structure (structure IIIC) used for output variable deviation (⊙) associated with a node on NFBL.

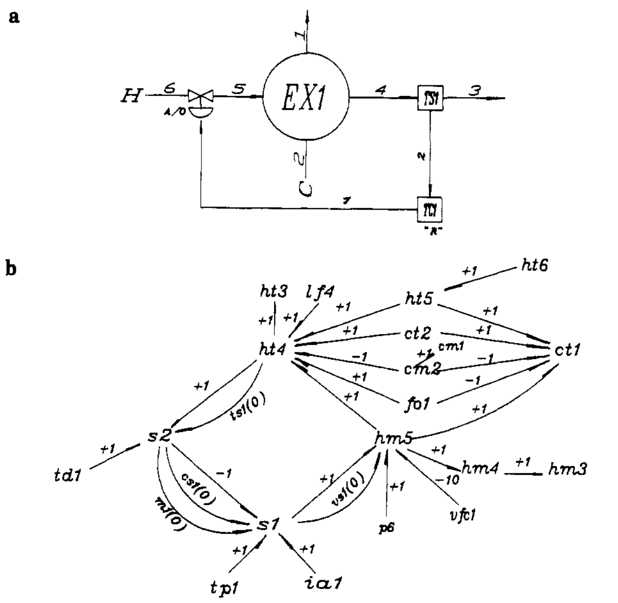


Figure 10. (a) Simplified process flow diagram of example 1. (b) Digraph model of example 1.

In this figure, TS_1 represents a temperature sensor, TC_1 is a reverse-action temperature controller, and the control valve V is air to open (A/O).

The system digraph has been drawn accordingly in Figure 10b. The physical meanings of the symbols used in this digraph can be found in the Nomenclature at the end of this paper. "$hm_5(-1)$", i.e. a moderate decrease in the flow rate of hot stream H before entering the heat exchanger, is used as the top event of this example. Two different fault trees can be obtained by implementing the conventional operator (fault tree A in Figure 11a) and the proposed structure III (fault tree B in Figure 11b). Their respective minimum cut sets are

tree A

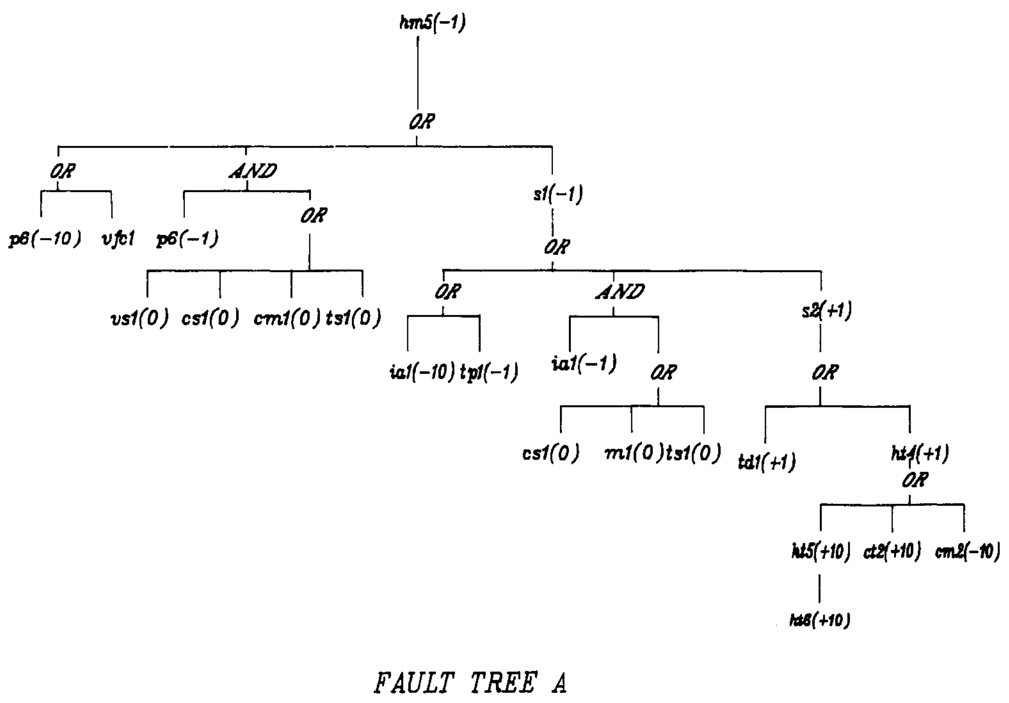
$\{p_6(-10)\}$, $\{vfc_1\}$, $\{tp_1(-1)\}$, $\{ia_1(-10)\}$, $\{td_1(+1)\}$, $\{ct_2(+10)\}$, $\{cm_2(-10)\}$, $\{ht_6(+10)\}$, $\{hm_6(-1),vs_1(0)\}$, $\{hm_6(-1),cs_1(0)\}$, $\{hm_6(-1),m_1(0)\}$, $\{hm_6(-1),ts_1(0)\}$, $\{ia_1(-1),cs_1(0)\}$, $\{ia_1(-1),m_1(0)\}$, $\{ia_1(-1)ts_1(0)\}$

tree B

$\{p_6(-10)\}$, $\{vfc_1\}$, $\{tp_1(-1)\}$, $\{ia_1(-10)\}$, $\{td_1(+1)\}$, $\{ct_2(+1)\}$, $\{cm_2(-1)\}$, $\{ht_6(+1)\}$, $\{lf_4\}$, $\{fo_1\}$, $\{hm_6(-1),vs_1(0)\}$, $\{hm_6(-1),cs_1(0)\}$, $\{hm_6(-1),m_1(0)\}$, $\{hm_a(-1),ts_1(0)\}$, $\{ia_1(-1), cs_1(0)\}$, $\{ia_1(-1),m_1(0)\}$, $\{ia_1(-1),ts_1(0)\}$

On the basis of the analysis presented in Table III, one can see clearly that top event $hm_5(-1)$ can be the result of fo_1 (moderate fouling in exchanger EX_1) or lf_4 (a small local fire near line 4). These two causes can only be found in the cut sets of fault tree B. One can also conclude from Table IV that the disturbance $ct_2(+10)$, $cm_2(-10)$, or $ht_6(+10)$ produces a negative deviation in hm_5 of magnitude 10, i.e. a drastic decrease in flow rate hm_5 to a level close to zero. They should not be considered as cut sets, since their mutually exclusive events, i.e. $ct_2(+1)$, $cm_2(-1)$, and $ht_6(+1)$, are more appropriate causes of $hm_5(-1)$ (see Table III).

Finally, it should be pointed out that structure III can also be used to develop fault trees corresponding to *process* NFBLs. As mentioned repeatedly in this paper, any disturbance that enters a NFBL creates two opposite effects on the loop variable associated with the entry node. In the case of the control loop, the two effects cancel out due to the regulatory action. The pattern of deviations in the loop variables as a result of local disturbance can then be a



b

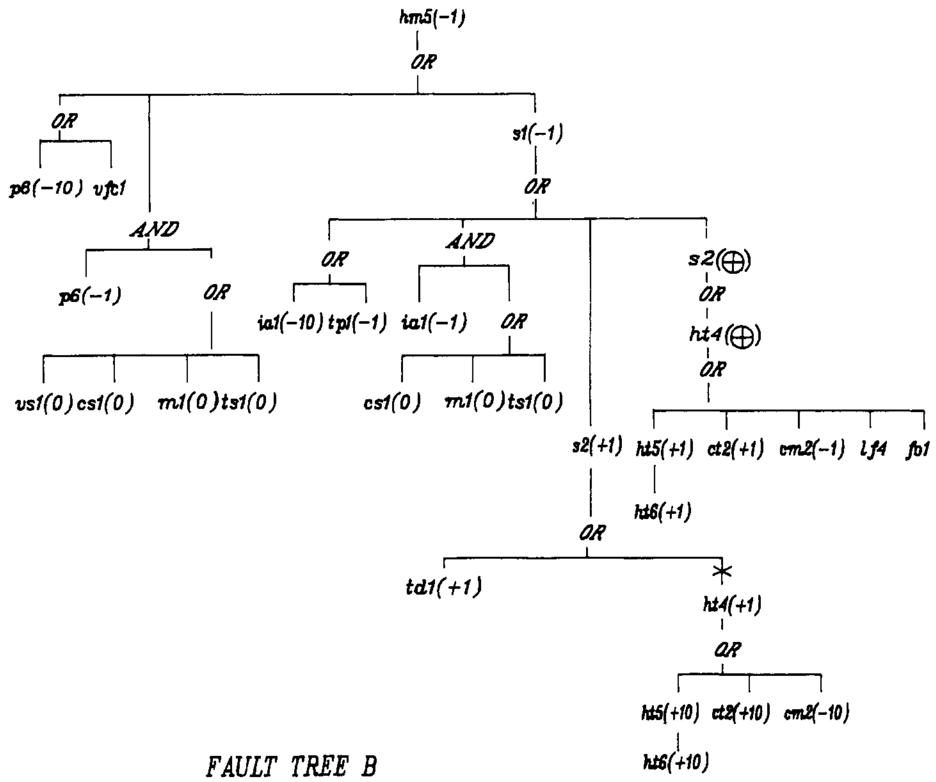


Figure 11. (a) Fault tree obtained by the conventional operator for the top event $hm_5(-1)$ in example 1. (b) Fault tree obtained by structure III for the top event $hm_5(-1)$ in example 1.

evaluated easily. However, if a process loop is considered, the net effect of local disturbance is unknown without further process information. Thus, although the same structure III can be utilized in synthesizing fault trees, the patterns of deviations in loop variables similar to Table III and Table IV must be determined in advance on a case-by-case basis using additional quantitative data of the system.

Example 2. Considered here is the simple process of a storage tank with input and output liquid flows (Figure 12a). The corresponding digraph is presented in Figure 12b. In this example, the disturbances of magnitude 10 are assumed to be improbable. The patterns of deviations in loop variables corresponding to disturbances of magnitude 1 can be established from simple analysis of the system and a table similar to Table III can be obtained (Table V).

Table V. Patterns of Variable Deviations in a Process FBL

disturbance	L	F_2
$F_1(+1)$	+1	+1
$X_2(+1)$	−1	⊕

Based on this information, fault trees can be easily constructed using structure III. Notice that, in this case, the part labeled (c) should be considered for every loop variable and the values of the disturbances marked (e) and

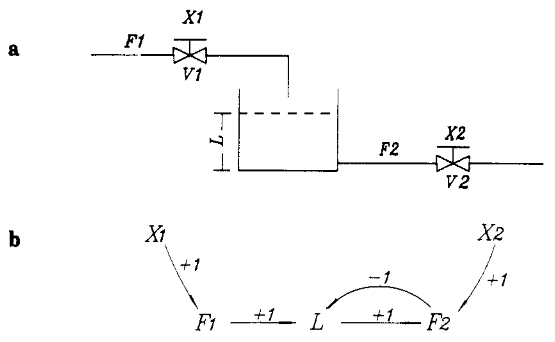


Figure 12. (a) Simplified process flow diagram of example 2. (b) Digraph model of example 2.

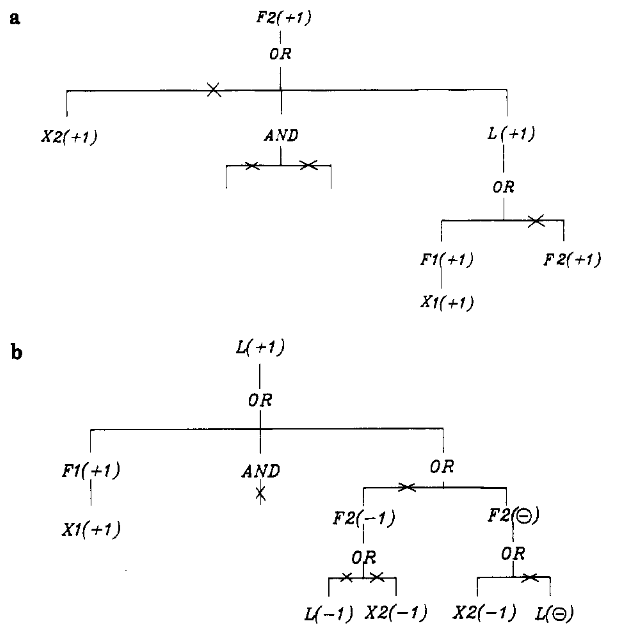


Figure 13. (a) Fault tree obtained by structure III for the top event $F_2(+1)$ in example 2. (b) Fault tree obtained by structure III for the top event $L(+1)$ in example 2.

(f) must be consistent with the patterns summarized in Table V. The fault trees corresponding to the top events $L(+1)$ and $F_2(+1)$ are presented in Figure 13a,b, respectively.

Loop Identification and Classification Algorithm

From the above discussions, one can see that it is necessary to locate all the control and process loops in a system digraph before the construction of fault trees. Since the digraph models of almost all realistic systems are very complex, it is extremely difficult to accomplish this task manually. Thus, there is a need for developing computer algorithms to identify and classify the FFLs and NFLs in digraphs.

There are a number of techniques available in the literature which can be implemented for similar purposes; e.g. see Tarjan (1972) and Deo (1974). The former developed methods for identifying strongly-connected components in directed graphs and biconnected components in undirected graphs. The latter proposed a loop-searching procedure based on the concept of a spanning tree. Notice that the FFLs and FBLs should be treated differently in the process of synthesizing fault trees. Thus, all the loops need to be not only identified but also classified according to their structural characteristics. Since none of the above approaches satisfy these requirements, an algorithm has been developed in this study to meet our needs. The detailed procedure of this algorithm is presented elsewhere (Hwang, 1991), and a computer program IDLOOP has been coded accordingly.

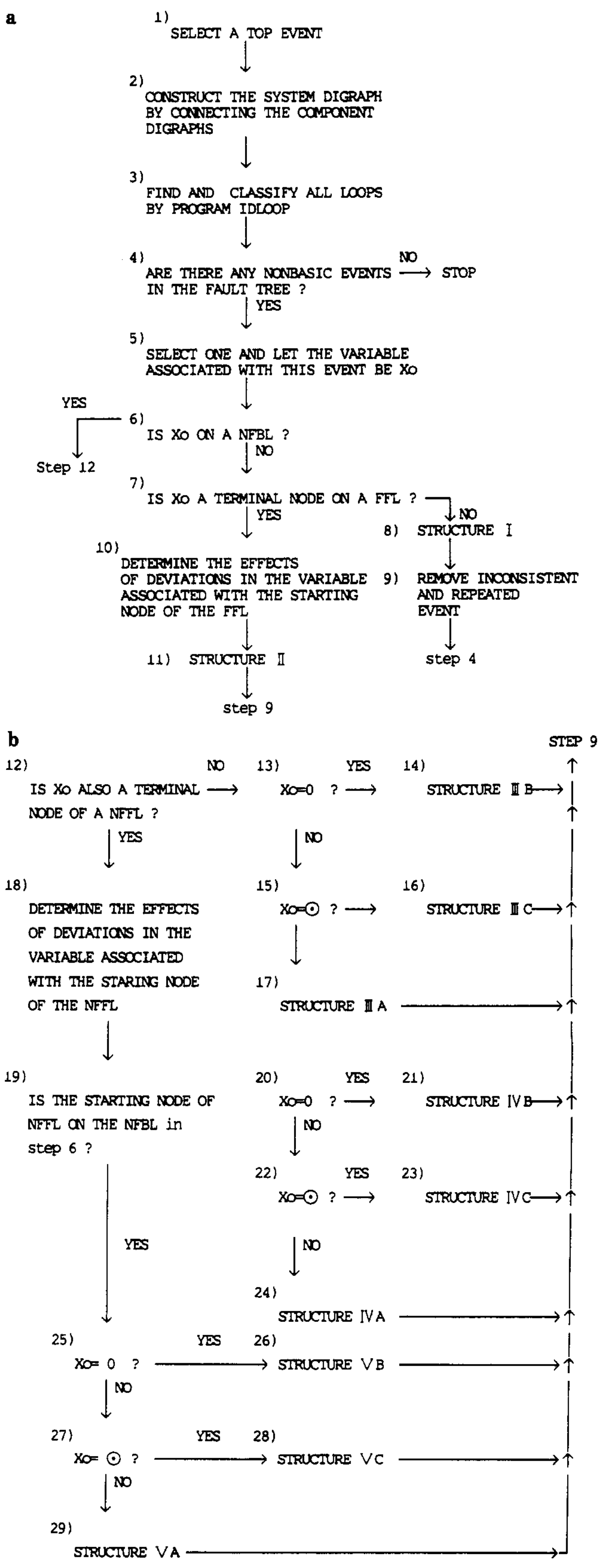

a
1) SELECT A TOP EVENT
2) CONSTRUCT THE SYSTEM DIGRAPH BY CONNECTING THE COMPONENT DIGRAPHS
3) FIND AND CLASSIFY ALL LOOPS BY PROGRAM IDLOOP
4) ARE THERE ANY NONBASIC EVENTS IN THE FAULT TREE ?
NO
STOP
YES
5) SELECT ONE AND LET THE VARIABLE ASSOCIATED WITH THIS EVENT BE Xo
YES
6) IS Xo ON A NFBL ?
Step 12
NO
7) IS Xo A TERMINAL NODE ON A FFL ?
YES
NO
8) STRUCTURE I
10) DETERMINE THE EFFECTS OF DEVIATIONS IN THE VARIABLE ASSOCIATED WITH THE STARTING NODE OF THE FFL
9) REMOVE INCONSISTENT AND REPEATED EVENT
11) STRUCTURE II
step 4
step 9
b
STEP 9
12) IS Xo ALSO A TERMINAL NODE OF A NFFL ?
NO
13) Xo=0 ?
YES
14) STRUCTURE III B
YES
NO
18) DETERMINE THE EFFECTS OF DEVIATIONS IN THE VARIABLE ASSOCIATED WITH THE STARING NODE OF THE NFFL
15) Xo=⊙ ?
16) STRUCTURE III C
17) STRUCTURE III A
19) IS THE STARTING NODE OF NFFL ON THE NFBL in step 6 ?
20) Xo=0 ?
YES
21) STRUCTURE IV B
NO
22) Xo=⊙ ?
YES
23) STRUCTURE IV C
YES
NO
24) STRUCTURE IV A
25) Xo= 0 ?
YES
26) STRUCTURE V B
NO
27) Xo= ⊙ ?
YES
28) STRUCTURE V C
NO
29) STRUCTURE V A

Figure 14. Flow chart of the modified fault-tree synthesis algorithm: (a) part 1; (b) part 2.

Fault-Tree Synthesis Procedure

The fault-tree synthesis procedure presented here is a modified version of the Lapp–Powers algorithm which

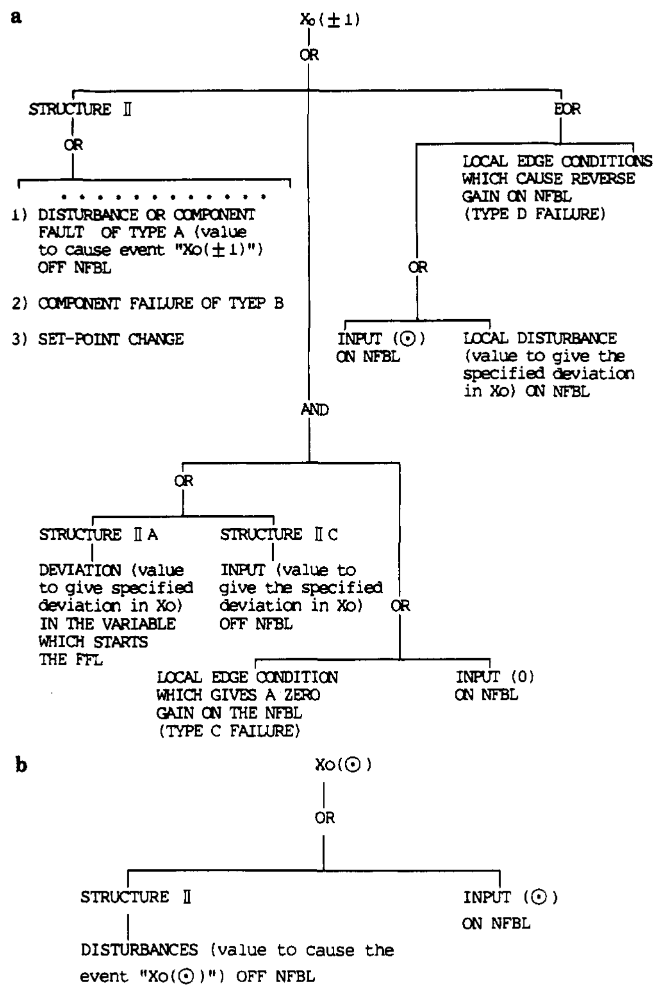

a
Xo(±1)
OR
STRUCTURE II
OR
EOR
LOCAL EDGE CONDITIONS WHICH CAUSE REVERSE GAIN ON NFBL (TYPE D FAILURE)
1) DISTURBANCE OR COMPONENT FAULT OF TYPE A (value to cause event "Xo(±1)") OFF NFBL
2) COMPONENT FAILURE OF TYEP B
3) SET-POINT CHANGE
OR
INPUT (⊙) ON NFBL
LOCAL DISTURBANCE (value to give the specified deviation in Xo) ON NFBL
AND
OR
STRUCTURE II A
DEVIATION (value to give specified deviation in Xo) IN THE VARIABLE WHICH STARTS THE FFL
STRUCTURE II C
INPUT (value to give the specified deviation in Xo) OFF NFBL
OR
LOCAL EDGE CONDITION WHICH GIVES A ZERO GAIN ON THE NFBL (TYPE C FAILURE)
INPUT (0) ON NFBL
b
Xo(⊙)
OR
STRUCTURE II
DISTURBANCES (value to cause the event "Xo(⊙)") OFF NFBL
INPUT (⊙) ON NFBL

Figure 15. (a) Substructure IVA. (b) Substructure IVC.

incorporates the additional features developed in this work. A simplified flow chart of this procedure is provided in Figure 14.

In this procedure, structure IV and structure V are assembled from the substructures of structure II and structure III. If the terminal node x_0 of a FFL is located on a NFBL but the starting node is off the feedback loop, the faults propagated from the nodes of the FFL to x_0 can be regarded as off-NFBL inputs in structure III. In such a case, structure IV presented in Figure 15a,b should be employed for developing the fault tree. Notice that since substructure IVB is the same as substructure IIIB, it is not repeated here. On the other hand, if the starting node of the FFL is also on NFBL in the above situation, the corresponding fault-tree should be constructed according to structure V, which is presented in Figure 16a–c.

Notice that the flow chart in Figure 14 is far from exhaustive. This procedure does not include a number of cases which may occur in digraphs with more complex loops. For example, it may happen that a node x_0 is both the terminal node of *multiple* FFLs and a node on NFBL and, also, the starting node of one of the FFLs is on the same NFBL. The corresponding fault-tree structure can be developed using the same principles presented in this paper. For the sake of brevity, extensions of this type are excluded. Another case which cannot be handled by the proposed procedure is when the node x_0 is simultaneously on two or more NFBLs, e.g. a cascade control loop. However, since this problem has already been addressed in the literature (Andow, 1980; Powers and Lapp, 1987), the corresponding fault-tree synthesis techniques are also omitted.

Application Examples

Again the system described in example 1 is considered. It has been found that although somewhat different fault trees can be generated by two alternate approaches (i.e. the conventional operators and the fault-tree structures developed in this study), the minimum cut sets corresponding to the top event $ht_3(+1)$ remain the same. Thus, the results related to the this particular top event are not reported here. Instead, the more interesting case of developing a fault tree with the top event $ct_1(-1)$ is studied. One may argue that the development of a fault tree corresponding to deviations in ct_1 does not make sense realistically. However, in a complex system, the event $ct_1(-1)$ can be an intermediate event in a larger tree. Some examples of such complex systems can be found in Figure 18a–c. Under these situations, the synthesis of a correct fault tree for ct_1 becomes quite important.

By making use of the computer program IDLOOP, the FBLs and FFLs in example 1 can be identified:

FBLs

$$hm_5 \leftarrow s_1 \leftarrow s_2 \leftarrow ht_4 \leftarrow hm_5 \qquad \text{(I)}$$

FFLs

$$\begin{pmatrix} ct_1 \leftarrow ht_5 \\ ct_1 \leftarrow hm_5 \leftarrow s_1 \leftarrow s_2 \leftarrow ht_4 \leftarrow ht_5 \end{pmatrix} \qquad \text{(II)}$$

$$\begin{pmatrix} ct_1 \leftarrow ct_2 \\ ct_1 \leftarrow hm_5 \leftarrow s_1 \leftarrow s_2 \leftarrow ht_4 \leftarrow ct_2 \end{pmatrix} \qquad \text{(III)}$$

$$\begin{pmatrix} ct_1 \leftarrow cm_2 \\ ct_1 \leftarrow hm_5 \leftarrow s_1 \leftarrow s_2 \leftarrow ht_4 \leftarrow cm_2 \end{pmatrix} \qquad \text{(IV)}$$

$$\begin{pmatrix} ct_1 \leftarrow fo_1 \\ ct_1 \leftarrow hm_5 \leftarrow s_1 \leftarrow s_2 \leftarrow ht_4 \leftarrow fo_1 \end{pmatrix} \qquad \text{(V)}$$

On the basis of the system P&ID (Figure 10a) and the gains specified in the corresponding digraph (Figure 10b), one can be sure that loop I is a control NFBL, loops II–IV are process NFFLs, and loop V is a process PFFL. From the system P&ID one can see that the net effect of loop IV should be negative since the heat exchanged in the system is maintained constant by the control action. The net effects of loops II and III, on the other hand, cannot be determined without further analysis of the quantitative process information. It is assumed that, for this particular system under consideration, the net effect of loop II is positive and that of loop III is negative. On the basis of these assumptions, the fault tree presented in Figure 17 can be obtained by following the proposed synthesis procedure. In this example, another fault tree has also been constructed by using the conventional operators. These two different fault trees can be analyzed by comparing their respective cut sets.

Common cut sets: $\{fo_1\}$, $\{vfc_1\}$, $\{ia_1(-10)\}$, $\{tp_1(-1)\}$, $\{td_1(+1)\}$, $\{ht_6(-1)\}$, $\{cm_2(+1)\}$, $\{p_6(-10)\}$, $\{p_6(-1),vs_1(0)\}$, $\{p_6(-1),cs_1(0)\}$, $\{p_6(-1),m_1(0)\}$, $\{p_6(-1),ts_1(0)\}$, $\{ia_1(-1),cs_1(0)\}$, $\{ia_1(-1),m_1(0)\}$, and $\{ia_1(-1),ts_1(0)\}$.

Cut sets which can only be obtained by the conventional operators:

(1) $\{ct_2(-1)\}$. This is a mistake. From our assumption for loop II, the net effect of a change in variable ct_1 to ct_2 should be negative. Thus, the correct cause of event $ct_1(-1)$ should be $ct_2(+1)$.

(2) $\{cm_2(-10)\}$, $\{ht_6(+10)\}$, **and** $\{ct_2(+10)\}$. Notice that they are contradictory with $\{cm_2(+1)\}$, $\{ht_6(-1)\}$, and $\{ct_2(-1)\}$, which are the cut sets of the same fault tree. These results are caused by implementing the conventional approach for developing fault trees for the terminal nodes of NFFLs. Also, on the basis of the analysis of process

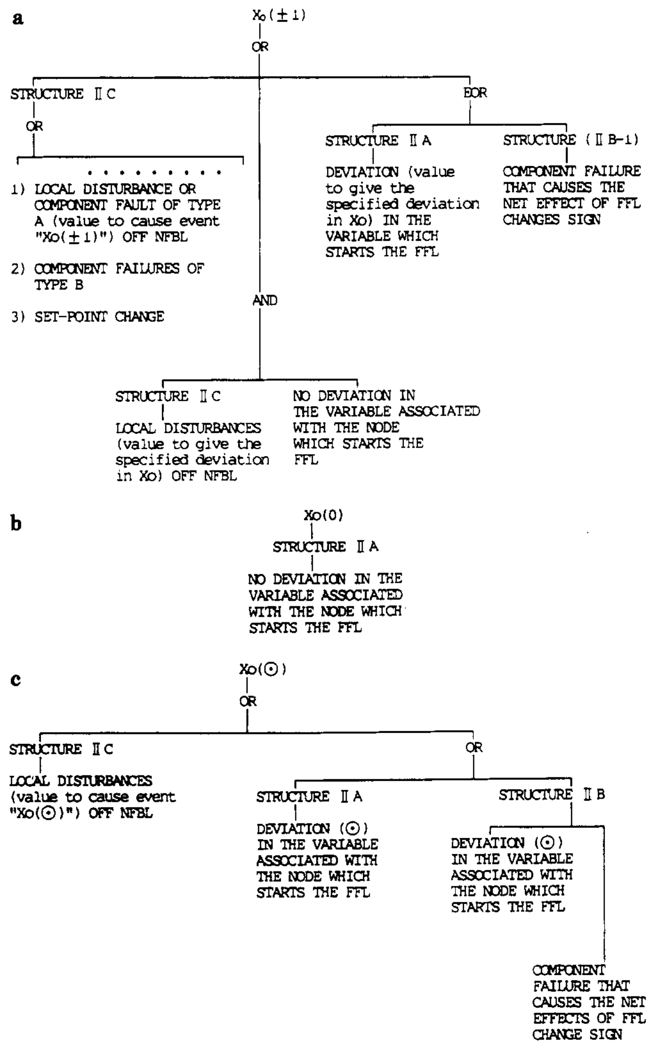

a
Xo(±1)
OR
STRUCTURE Ⅱ C
OR
1) LOCAL DISTURBANCE OR COMPONENT FAULT OF TYPE A (value to cause event "Xo(±1)") OFF NFBL
2) COMPONENT FAILURES OF TYPE B
3) SET-POINT CHANGE
EOR
STRUCTURE Ⅱ A
DEVIATION (value to give the specified deviation in Xo) IN THE VARIABLE WHICH STARTS THE FFL
STRUCTURE (Ⅱ B-1)
COMPONENT FAILURE THAT CAUSES THE NET EFFECT OF FFL CHANGES SIGN
AND
STRUCTURE Ⅱ C
LOCAL DISTURBANCES (value to give the specified deviation in Xo) OFF NFBL
NO DEVIATION IN THE VARIABLE ASSOCIATED WITH THE NODE WHICH STARTS THE FFL
b
Xo(0)
STRUCTURE Ⅱ A
NO DEVIATION IN THE VARIABLE ASSOCIATED WITH THE NODE WHICH STARTS THE FFL
c
Xo(⊙)
OR
STRUCTURE Ⅱ C
LOCAL DISTURBANCES (value to cause event "Xo(⊙)") OFF NFBL
OR
STRUCTURE Ⅱ A
DEVIATION (⊙) IN THE VARIABLE ASSOCIATED WITH THE NODE WHICH STARTS THE FFL
STRUCTURE Ⅱ B
DEVIATION (⊙) IN THE VARIABLE ASSOCIATED WITH THE NODE WHICH STARTS THE FFL
COMPONENT FAILURE THAT CAUSES THE NET EFFECTS OF FFL CHANGE SIGN

Figure 16. (a) Substructure VA. (b) Substructure VB. (c) Substructure VC.

information, the net effects of cm_2 and ht_6 to ct_1 have been determined to be −1 and +1, respectively. Naturally, the

Table VI. Digraph Characteristics of the Application Examples (Problems I–III)

	I	II	III
no. of nodes	26	65	58
no. of edges	35	91	81
no. of control FBLs	1	3	3
no. of control FFLs	0	0	0
no. of process FBLs	1	0	1
no. of process FFLs	20	46	68
total no. of paths in FFLs	59	238	420

events $cm_2(-10)$ and $ht_6(+10)$ cannot be the causes of $ct_1(-1)$. Finally, from the assumption for loop II, the event $ct_2(+1)$ alone can produce the net effect of a moderate decrease in ct_1. Thus, the cut set $\{ct_2(+10)\}$ becomes unnecessary in this situation. In summary, these erroneous cut sets are generated mainly due to the fact that the conventional operator of NFBLs does not consider the patterns of deviations created by "controllable" disturbances such as those listed in Table III.

Cut sets which can only be obtained by the proposed procedure:

(1) $\{lf_4\}$. If a local fire occurs near line 4, then the manipulated variable hm_5 must be decreased to maintain ht_4 at a constant level (see Table III). As a result, the outlet temperature of the cold stream ct_1 should be lowered based on the digraph model in Figure 10b.

(2) $\{ct_2(+1)\}$. This is a direct result of the assumption for loop II.

(3) $\{ct_2(-1),vs_1(0)\}$, $\{ct_2(-1),cs_1(0)\}$, $\{ct_2(-1),m_1(0)\}$, and $\{ct_2(-1),ts_1(0)\}$. If a component fails and causes zero gain on the second path of loop II, i.e. $vs_1(0)$, $ct_1(0)$, $m_1(0)$, or $ts_1(0)$, then the effect of ct_2 to ct_1 is determined solely by the first path. Since the product of the edge gains on the first path is +1, the top event must be caused by $ct_2(-1)$ in this case.

In the addition to the above example, several more realistic problems have been studied. Three typical cases are presented in Figure 18a–c. Due to the limitation of space, their corresponding fault trees are not included in this paper. Detailed results of implementing the proposed procedure have been published elsewhere (Hwang, 1991). A summary of the corresponding system digraphs is

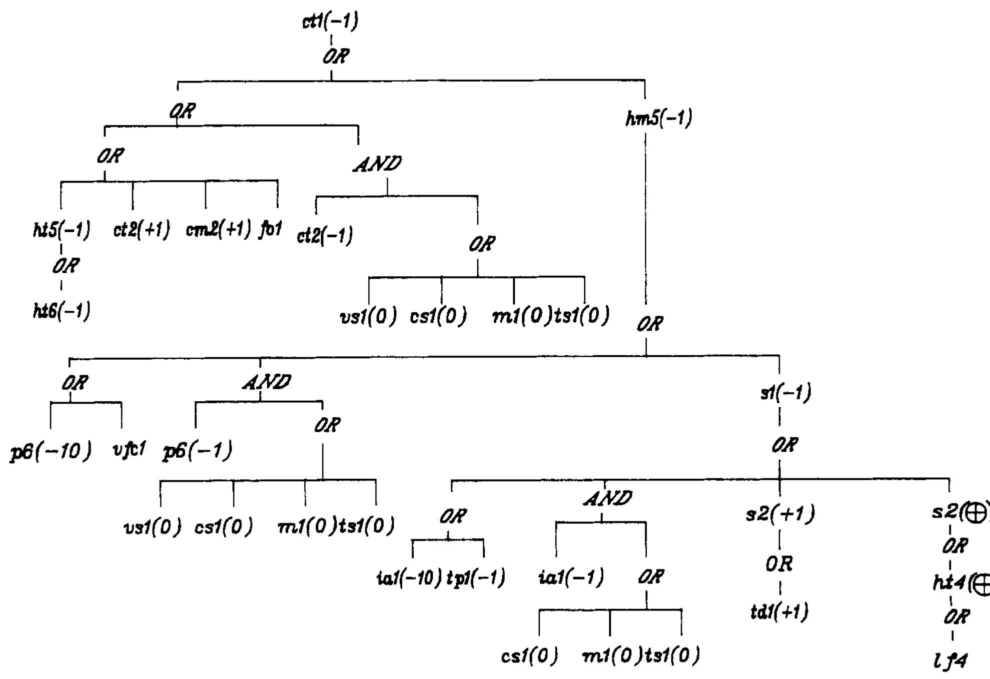


Figure 17. Fault tree obtained by the modified synthesis algorithm for the top event $ct_1(-1)$ in example 1.

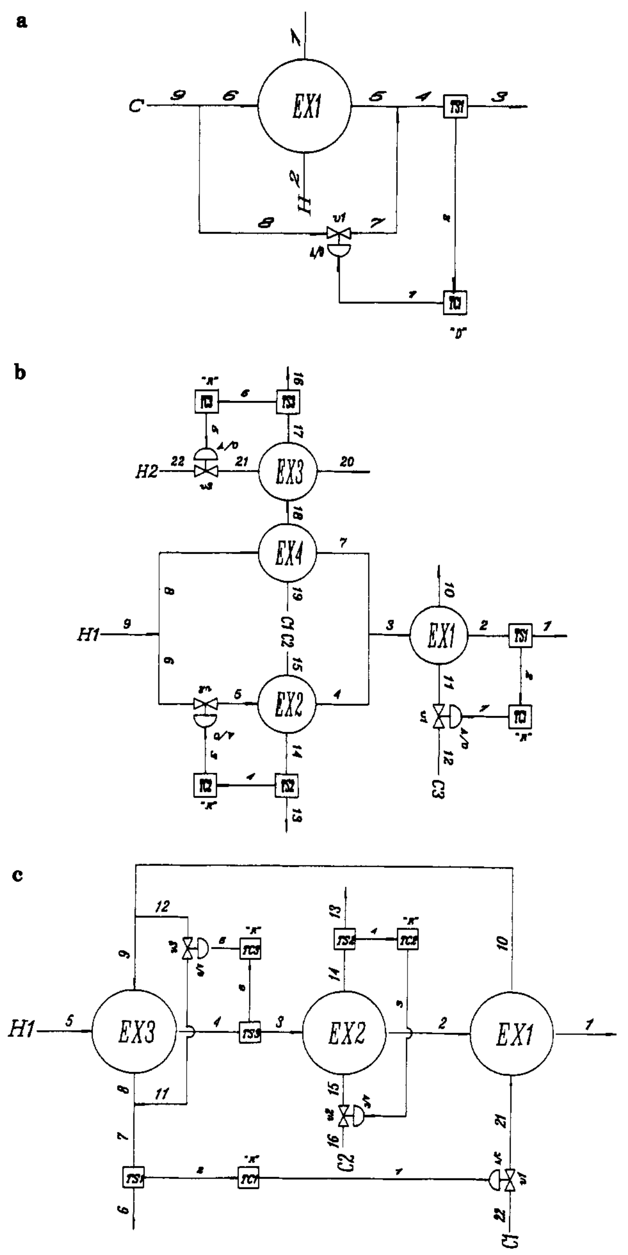


Figure 18. Simplified process flow diagrams of application examples: (a) problem I; (b) problem II; (c) problem III.

presented in Table VI just to show the complexity of these problems. Notice that the total numbers of paths listed in this table are calculated by adding the numbers of paths in all FFLs of the system. Thus, some of the paths may be counted repeatedly.

Conclusions

From the previous discussions, it can be concluded that qualitative steady-state analysis of the fault propagation behaviors in process systems is indeed useful in developing improved fault-tree construction techniques. The generalized sub-tree structures proposed in this paper can be applied to systems with complex control and process loops. The results of implementing the suggested synthesis procedure to application examples show that the approach taken in this study is feasible and effective. Further, when compared with the fault trees built by the conventional operators, the trees obtained by our method are simpler (in the sense that repeated events are eliminated), more consistent (in the sense that mutually exclusive cut sets are avoided), and more comprehensive (in the sense that additional causes of the top events are identified) and, thus, should be more appropriate for the tasks of risk assessment.

Nomenclature

cm_1, cm_2 = mass flow rate of cold stream at pipelines 1 and 2, respectively
cr_1 = controller reversed (type D failure)
ct_1, ct_2 = temperature of the cold stream at pipelines 1 and 2, respectively
$cs_1(0)$ = controller stuck (type C failure)
fo_1 = moderate fouling in the exchanger (type A fault)
hm_3–hm_5 = mass flow rate of hot stream at pipelines 3–5, respectively
ht_3–ht_6 = temperature of hot stream at pipelines 3–6, respectively
ia_1 = instrument air pressure
lf_4 = a small local fire near pipeline 4 (type A fault)
$m_1(0)$ = controller on manual (type C failure)
p_6 = upstream pressure of the hot stream at pipeline 6
s_1 = controller output at signal line 1
s_2 = sensor output at signal line 2
$td_1(+1)$ = temperature sensor failing high (type B failure)
$tp_1(-1)$ = set point changed to a lower value
$ts_1(0)$ = temperature sensor stuck (type C failure)
vfc_1 = control valve failing close (type B failure)
vr_1 = control valve reversed (type D failure)

Literature Cited

Allen, D. J. Digraphs and Fault Trees. *Ind. Eng. Chem. Fundam.* **1984**, *23*, 175.

Allen, D. J.; Rao, M. S. M. New Algorithms for the Synthesis and Analysis of Fault Trees. *Ind. Eng. Chem. Fundam.* **1980**, *19*, 79.

Andow, P. K. Difficulties in Fault Tree Synthesis for Process Plant. *IEEE Trans. Reliab.* **1980**, *R-29*, 2.

Andow, P. K. Fault Tree and Failure Analyses: Discrete State Representation Problems. *Trans. Inst. Chem. Eng.* **1981**, *59*, 125.

Andrews, J. D.; Morgan, J. M. Application of Digraph Method of Fault Tree Construction to Process Plant. *Reliab. Eng.* **1986**, *14*, 85.

Andrews, J. D.; Brennan, G. Application of the Digraph Method of Fault Tree Construction to a Complex Control Configuration. *Reliab. Eng. Syst. Saf.* **1990**, *28*, 357.

Camarda, P.; Corsi, F.; Trentadue, A. An Efficient Simple Algorithm for Fault Tree Automatic Synthesis from Reliability Graph. *IEEE Trans. Reliab.* **1978**, *R-27*, 215.

Chamow, M. F. Directed Graph Techniques for the Analysis of Fault Trees. *IEEE Trans. Reliab.* **1978**, *R-27*, 7.

Chang, C. C.; Yu, C. C. On-Line Fault Diagnosis Using the Signed Directed Graph. *Ind. Eng. Chem. Res.* **1990**, *29*, 1290.

Cummings, D. L.; Lapp, S. A.; Powers, G. J. Fault Tree Synthesis From a Directed Graph Model for a Power Distribution Network. *IEEE Trans. Reliab.* **1983**, *R-32*, 140.

Deo, N. *Graph Theory with Applications to Engineering and Computer Science*; Prentice-Hall: Englewood Cliffs, NJ, 1974.

Fussell, J. B. Synthetic Fault Tree Model-A Formal Methodology for Fault Tree Construction; Report *ANCR 1098*; National Reactor Testing Station, Aerojet Nuclear Co.: Idaho Falls, ID, 1973.

Galluzzo, M.; Andow, P. K. Failures in Control Systems. *Reliab. Eng.* **1984**, *7*, 193.

Hwang, H. C. New Developments in the Digraph-Based Approach for Fault Tree Analysis. M.S. Thesis, National Cheng Kung University, Tainan, Taiwan, ROC, 1991.

Iri, M.; Aoki, K.; O'Shima, E.; Matsuyama, H. An Algorithm for Diagnosis of System Failures in the Chemical Process. *Comput. Chem. Eng.* **1979**, *3*, 489.

Kelly, B. E.; Lees, F. P. The Propagation of Faults in Process Plants: 1. Modelling of Fault Propagation. *Reliab. Eng.* **1986a**, *16*, 3.

Kelly, B. E.; Lees, F. P. The Propagation of Faults in Process Plants: 2. Fault Tree Synthesis. *Reliab. Eng.* **1986b**, *16*, 39.

Kelly, B. E.; Lees, F. P. The Propagation of Faults in Process Plants: 3. An Interactive, Computer-Based Facility. *Reliab. Eng.* **1986c**, *16*, 63.

Kelly, B. E.; Lees, F. P. The Propagation of Faults in Process Plants: 4. Fault Tree Synthesis of a Pump System Changeover Sequence. *Reliab. Eng.* **1986d**, *16*, 87.

Khan, A. R.; Hunt, A. The Propagation of Faults in Process Plants: Integration of Fault Propagation Technology into Computer-aided Design. *Inst. Chem. Eng. Symp. Ser.* **1989**, *114*, 35.
Kramer, M. A.; Palowitch, B. L., Jr. A Rule-Based Approach to Fault Diagnosis Using the Signed Directed Graph. *AIChE J.* **1987**, *33*, 1067.
Kumamoto, H.; Henley, E. J. Safety and Reliability Synthesis of Systems with Control Loops. *AIChE J.* **1979**, *20*, 376.
Kumamoto, H.; Henley, E. J.; Inoue, K. Signal-Flow-Based Graphs for Failure-Mode Analysis of Systems with Control Loops. *IEEE Trans. Reliab.* **1981**, *R-30*, 110.
Lambert, H. E. Comments on the Lapp-Powers 'Computer-Aided Synthesis of Fault Trees.' *IEEE Trans. Reliab.* **1979**, *R-28*, 6.
Lapp, S. A.; Powers, G. J. Computer-Aided Synthesis of Fault Trees. *IEEE Trans. Reliab.* **1977**, *R-26*, 2.
Lapp, S. A.; Powers, G. J. Update of Lapp-Powers Fault-Tree Synthesis Algorithm. *IEEE Trans. Reliab.* **1979**, *R-28*, 12.
Lee, W. S.; Grosh, D. L.; Tillman, F. A.; Lie, C. H. Fault Tree Analysis, Method, and Applications-A Review. *IEEE Trans. Reliab.* **1985**, *R-34*, 194.
Oyeleye, O. O.; Kramer, M. A. Qualitative Simulation of Chemical Process Systems: Steady State Analysis. *AIChE J.* **1988**, *34*, 1441.
Powers, G. J.; Lapp, S. A. *A Short Course on Risk and Reliability Assessment by Fault Tree Analysis*; Post College Professional Education Center, Carnegie-Mellon University: Pittsburgh, PA, 1987.
Qian, D. Q. An Improved Method for Fault Location of Chemical Plants. *Comput. Chem. Eng.* **1990**, *14*, 41.
Salem, S. L.; Apostolakis, G. E.; Okrent, D. A New Methodology for the Computer-aided Construction of Fault Trees. *Ann. Nucl. Energy* **1977**, *4*, 417.
Shaeiwitz, J. A.; Lapp, S. A.; Powers, G. J. Fault Tree Analysis of Sequential Systems. *Ind. Eng. Chem. Process Des. Dev.* **1977**, *16*, 529.
Shafaghi, A.; Andow, P. K.; Lees, F. P. Fault Tree Synthesis Based on Control Loop Structure. *Chem. Eng. Res. Des.* **1984**, *62*, 101.
Shiozaki, J.; Matsuyama, H.; O'Shima, E.; Iri, M. An Improved Algorithm for Diagnosis of System Failures in the Chemical Process. *Comput. Chem. Eng.* **1985**, *9*, 285.
Tarjan, R. Depth-First Search and Linear Graph Algorithms. *SIAM J. Comput.* **1972**, *1*, 146.

Received for review August 15, 1991
Revised manuscript received February 5, 1992
Accepted March 4, 1992

Avoiding Accumulation of Trace Components

Sanjay K. Joshi and James M. Douglas*
Department of Chemical Engineering, University of Massachusetts, Amherst, Massachusetts 01003

The accumulation of trace components in a recycle loop can cause a process to become inoperable. To prevent this problem, it is often necessary to add additional exit points to the flowsheet, at the conceptual stage of a process design, in order to remove the trace components. In this paper we present a systematic procedure for identifying exit points which are needed in order to ensure that the design will be operable.

Introduction

Just about every company has had the unfortunate experience during the startup of a new process to observe the continual buildup of trace components in a recycle loop, so that eventually the process fails to meet its design specifications. It is significantly more expensive to shut down the process, empty it, purge the system, add new exit points, and restart the process than it is to add additional exit points before construction. For this reason, many experienced designers add an exit point in each recycle loop.

However, the addition of an exit point in each recycle stream has the disadvantage that the exit points that are used during operation to prevent the accumulation of trace components often will cause waste treatment problems, which is in conflict with the new EPA priority of eliminating pollution problems at the source. Similarly, the exit points that are not used during operation correspond to wasted capital required to install the exit point. Hence, it would be advantageous to have a systematic procedure available that would indicate when additional exit points are needed, and to be able to estimate the increased recycle flows caused by trace components. Another advantage of a procedure of this type is that it can be automated. We present a procedure of this type below.

By definition, a trace component is present in only small amounts in an input stream or produced in a reactor, so that trace components have a negligible effect on the overall material balances. However, these trace components can accumulate in recycle loops. If there is no way that the trace component can exit from the recycle loop, it will continue to accumulate until the process cannot meet the original design specifications. Even if the trace component can build up to a level where it can leave the recycle loop, if the design of the equipment in the recycle loop is not based on the increased recycle flow, the process might not be able to meet the overall design specifications.

Table I. Primary Classification of Recycle Loops

1.	reactant recycle loops
2.	separation unit recycle loops
3.	separation system recycle loops

Since we are primarily concerned with the accumulation of trace components in recycle loops, we can simplify the problem if we classify the types of recycle loops that are common in flowsheets. The primary classifications are given in Table I. Separation unit recycle loops can then be classified as absorber recycles, extraction recycles, cake washing recycles, etc. Simple, common-sense, heuristics can be used to identify the need for exit points for reactor recycle loops and separation unit recycle loops. However, separation system recycle loops, i.e., where there are interconnected separation unit recycle loops in a separation system, are much more complicated. Hence, we present a new notation for treating these problems.

Previous Work

Douglas (1985, 1988) described a hierarchical synthesis procedure for the conceptual design of chemical processes. However, while developing alternative flowsheets, the synthesis procedure does not consider the presence of trace component impurities. There does not seem to have been any previous attempt to develop a systematic procedure for adding new exit points to a flowsheet in order to avoid