Difficulties in Fault-Tree Synthesis for Process Plant

P.K. Andow
University of Technology, Loughborough

Key Words—Fault-tree synthesis, Control loops

Reader Aids—
Purpose: Explore problem areas
Special math needed: None
Results useful to: Reliability engineers, Designers

Summary & Conclusions—**This paper identifies a number of related difficulties, some of which are still unsolved. Attention is drawn to failings in the type of pressure-flow model commonly used in the literature. Difficulties also exist when published algorithms are applied to control loops. These are illustrated for simple and cascade control applications and discussed in some detail. Eight general conclusions are:**

1. The concept of 2-way flow of information in failure models is important in certain situations, e.g., fluid flow.

2. The accuracy of failure models is generally low. This reflects the fact that much of the effort expended in systematic failure analyses has been heavily oriented towards algorithms.

3. Models used in failure analyses do not have to be comprehensive. Only the credible set of events is needed.

4. No always-satisfactory algorithm has been published for fault-tree synthesis where control loops are encountered.

5. The control loop problem is inextricably interlinked with the general difficulty that fault-tree methodology is primarily oriented to binary systems where the time dimension can be ignored.

6. Fault-tree methodology uses simple models to approximate system failures. If these failures are complex then fault trees might not be suitable. The results of analyses involving complex failures must be treated with great care.

7. When fault-tree methodology is not completely suitable one ought to consider using a different technique altogether. The cause-consequence diagram might be appropriate since it can be used to study failure modes where time is important.

8. Algorithms must be carefully examined and properly validated before widespread use of computer-aided fault-tree synthesis is attempted. If this is not done, computer-aided synthesis will fall into disrepute.

I. INTRODUCTION

There has been a consistent movement towards improving the quality of process plant design. The reasons for this derive mainly from two sources:

1. For purely economic reasons there has been a trend towards larger plants. Many modern plants are single-stream designs. In order to reap the benefits of such designs the plants must be reliable.

2. For social and economic reasons there is an increasing interest in plant safety. An abnormal incident may cause severe damage to the plant in addition to exposing the employees, and possibly the public, to the risk of injury or death.

Reliability and safety studies can improve the plant design. Fault trees have been widely used as a tool in such studies. Various computer codes have been produced to reduce the large effort required to evaluate fault trees. These codes are concerned with one of two categories:

1. Numeric calculation of the top-event probability, given failure-data for the components. The quantification of plant reliability and availability can be a design criterion that the contractor has to fulfill. The fault-tree evaluation then is a documentation aid in addition to a design tool.

2. Finding the minimum combinations of failures that will cause the top-event. In a large reliable system there may be thousands of minimum cut sets. This type of evaluation brings the failure mechanism to the analysts' attention with a strong emphasis on system failures caused by 1- or 2-event cut sets.

In many studies both types of codes are used, i.e. the cut sets are found and then the system failure probability calculated. A common feature of both types of codes is that they are used to analyse a fault tree. More recently there has been considerable interest in the use of computer codes for fault-tree synthesis because it is a complex and time-consuming task.

Fussell [1] pioneered the work in this area with his Synthetic Tree Model (STM). Features were:

1. Primarily for electrical systems.
2. Logic models used as component transfer functions.
3. Discriminator flags used to ensure internal consistency.
4. Computer code produced.

Tompkins & Powers [2] concentrated primarily on defining a methodology for synthesis of process-plant fault trees. The basic theme was similar to Fussell's [1] except that a functional model was defined which showed the interactions between process variables. No computer code was produced.

Andow & Lees [3] also used a functional model to define and synthesize models for the related area of real-time analysis of process-plant alarms. Features were:

1. Component models were combined to create a network of nodes and links.
2. Each node represented a variable and each link showed how one variable effected another in terms of direction of change and time-lag.
3. Later refined to include magnitude of interactions.
4. Information-flow and process-flow treated as distinct but related properties of the system.
5. Computer code produced.

The network produced by the code was similar to a signal-flow diagram. For alarm analysis the network was reduced by eliminating nodes which represent non-measured variables. The final network normally contained multiple links between some nodes; these links represented information paths with different dynamic characteristics. This representation has obvious similarities with the fault tree.

0018-9529/80/0400-0002$00.75

Martin-Solis et al. [4] have reported further work using this type of model [3] and demonstrate the basic methodology to incorporate these 2-way information-flow models into a fault tree.

Apostolakis et al. [5] produced the Computer Automated Tree (CAT) code. Features are:

1. Decision table models contained multiple input and output states suitable for non-binary systems.
2. The code could synthesize trees containing AND gates. Earlier codes did not do this.

Lapp & Powers (L & P) [6] produced the Fault-Tree Synthesis (FTS) code. Features are:

1. A 2-step method based on constructing a fault-tree from a directed graph (digraph) which represents the system interactions. The digraph is produced by hand and is similar to the network used by Andow & Lees [3].
2. Multiple-state values considered for both nodes and links.
3. No direct account is taken of time, although L. & P. claim that sequences can be handled by the use of special models. The process models in the published work show information and process flows in the same direction only.
4. Automatic detection of feedback and feedforward loops and use of this information in the synthesis.

Taylor & Hollo [7] use algebraic component models to construct a Cause-Consequence Diagram (CCD). The CCD is the most comprehensive representation. The CCD method features:

1. Forward and backward development through time, giving a more complete picture of system failures—with a corresponding increase in complexity.
2. Applied to chemical, nuclear, and electrical systems.
3. Sequences of events considered.
4. Loops considered.

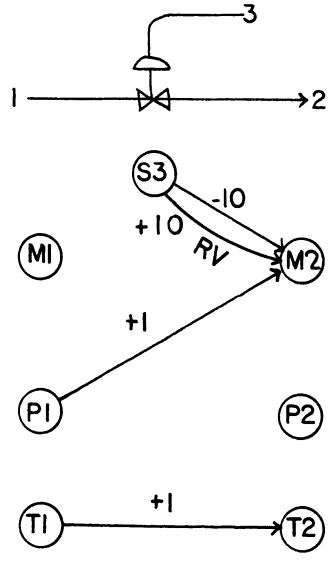
A number of other papers have been written commenting on various other aspects of the L. & P. algorithm. These are included in the references [8-12] for completeness.

I think that the work on fault propagation and its representation has reached a point where the basic methodology is reasonably well understood and tested. The area which has received less attention is the production of good failure models. Ideally the models would be independent of the synthesis method but, in practice, they are strongly interdependent.

II. INFORMATION FLOW AND MODEL DEFINITION

A. Simple Models

Consider a simple pipeline section which contains a valve. The pressure decays steadily along the pipeline. If the valve is closed then the pressure profile becomes quite different. If the valve closure is an event that can occur in a fault sequence then the system model must be capable of reflecting the fact that information must flow both upstream and downstream from the valve. A typical valve model from the literature is shown in Figure 1 in decision table form and in the form of a graph showing the information flow implied by the model.



in \ out	M2	T2
P1	+1	0
S3	-10	0
T1	0	+1
RV Valve Action	S3→M2 now +10	

Fig. 1. Valve Model

For convenience the L. & P. notation is used. Four features are important:

1. All of the information flows are in the same direction as the process flow.
2. The decision table defines the input and output variables. The model does not show how P1 would change if some downstream fault caused M2 to be zero.
3. The variables M1 and P2 (logically consistent with the use of M2 and P1) are not included at all.
4. Only 1 failure is shown.

The inclusion of only 1 failure mode does not necessarily indicate a poor model, in spite of the fact that a valve can have many other failure modes. Every model must have its set of credible events defined. As long as these are properly defined, and the model is used within its limitations, then all will be well. The model in Figure 1 would be a very poor general model because it does not reflect the normal propagation of pressure information from outlet to inlet. If used in a library of models for fault analysis it would only be useful if the analyst were certain beforehand that faults would always propagate in the same direction as process flows (e.g. in a fault-tree analysis, this would not be true if the causes of M2 being in a certain state were required). Since normal propagation is always a credible event general models must show all such effects.

B. General Models

Figure 2 shows a more general model which, for the purposes of comparison, still includes: a) only 1 failure originating

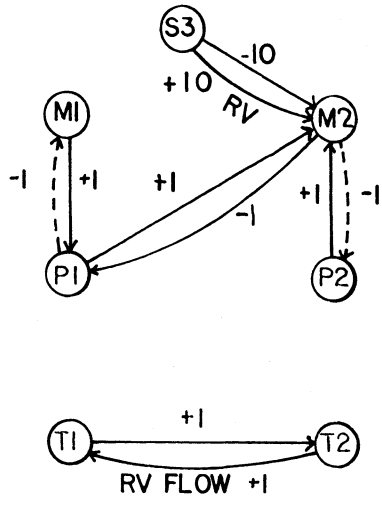


Fig. 2. More General Model

in the valve. b) information flow paths from outlet to inlet for the 3 primary variables—mass flow, pressure, and temperature. The model is not perfect, but it is more general and it is correspondingly more complex. Even the structure of the model is not unique. It could be argued that, for reverse-flow, the forward and backward paths linking M1 and P2 should be shown, but this is not so because the model is essentially an imperfect representation of the pressure-flow behaviour of the fluid at discrete points, whereas in reality these variables change continuously along the line. The model is however adequate for its purpose. In order to verify the model, simulations may be carried out using algebraic and differential equations containing the same information-path structure. If this is done for the valve/pipe, then transients can be obtained (for failures such as pipe break) which lead to reverse flow in parts of the line.

These transients are qualitatively correct. The quantitative results obtained obviously depend on flow coefficients and other factors which are system-dependent and cannot therefore be included in a general model. For the valve/pipe model the type of information required is limited to the simpler discrete categories HI, NORMAL, LO, NONE, REVERSE, etc. that are usually encountered in failure analyses. If more accurate failure information is needed (e.g. for a detailed study of cooling systems in a nuclear reactor) then the model is not sufficient. The model does show the dependence of T1 on T2 when the flow is reversed, but does not show the null case when input and output variables in the flow streams are not directly related because the valve is tightly shut.

C. Library-Model Conventions

The dashed information paths in Figure 2 are provided by other models also taken from a library of models. This illustrates a further desirable feature of general-purpose models; they should adhere to a set of conventions so that they are consistent with each other. Three useful conventions for flow modeling are:

1. Define all model streams as inputs or outputs in terms of the design-condition process-flows.
2. Use defining equations to set the pressure variable in process-flow input streams.
3. Use defining equations to set the mass-flow variable in process-flow output streams.

This is not the only set of conventions suitable for defining flows and pressures for this class of problems, but it is often sufficient. In common with other conventions it is not always applied, but care is then taken to mark the model.

III. CLOSED-LOOP INFORMATION PATHS

Closed-loop information paths present some difficulties in fault-tree analysis. Process plants frequently include negative feedback loops for control purposes. L. & P. [6] have given some examples of negative and positive feedback and feed-forward loops. Their approach is based on the use of special operators that are applied when an event is developed that lies on a loop. The FTS code searches the digraph model of the process in order to find and classify all loops. This approach has been demonstrated in the literature for a simple problem.

If this approach is applied to a digraph assembled from the type of model shown in Figure 2 then three difficulties arise:

1. Feedback loops appear in the digraph but they are not control loops in the usual sense. L. & P. published algorithms do not specifically consider this case although L. & P. recognize the existence of such loops.
2. The digraph contains loops within loops. The published algorithms refer to this problem but do not clearly define the solution.
3. Care must be taken in evaluating the results produced by applying fault-tree generation algorithms to control-loops.

This paper defines a modified L. & P. algorithm to handle the type of model shown in Figure 2. This modification is minor and overcomes difficulty #1. The algorithm is then applied to some simple examples. The examples show that difficulty #2 is easily overcome and illustrates difficulty #3.

IV. EXAMPLE #1

The very simple digraph shown in Figure 3 demonstrates the basis of the algorithm. Figure 3 could be handled by the published algorithms. The digraph represents a flow control loop; the controlled variable is FC, with unspecified external disturbances entering at each mode of the loop. Consider the top event FC(+1). Figure 4 shows the raw tree developed for this event where, for simplicity, the only credible disturbances are all moderate (i.e. + 1 or - 1 states) and the negative-feedback loop components are all correctly installed (i.e. no reverse components). Each event and gate in the tree is numbered in the order in which it was created. Comment boxes are included as an aid to understanding the logic used to build the tree. Figure 5 shows the same tree after being reduced by removing:

1. Normal condition events
2. Inconsistent events
3. Single-input gates
4. Event description boxes

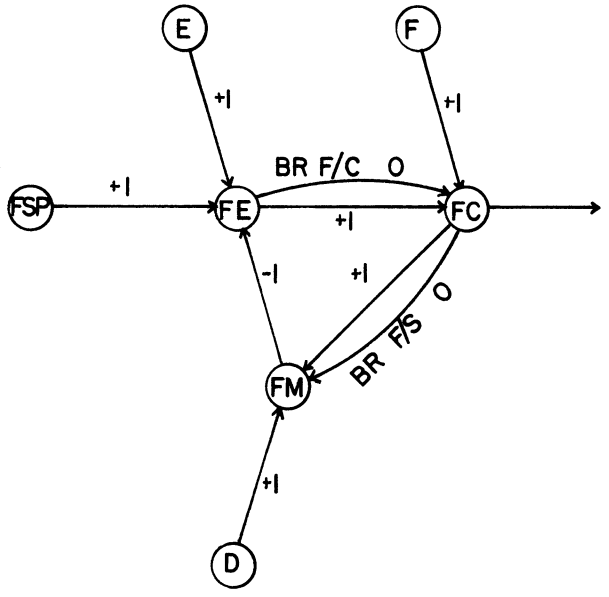


Fig. 3. Digraph for Flow Control Loop

The event and gate numbers of Figure 4 are retained for the purpose of comparison. The layout of the tree is similar for the same reason.

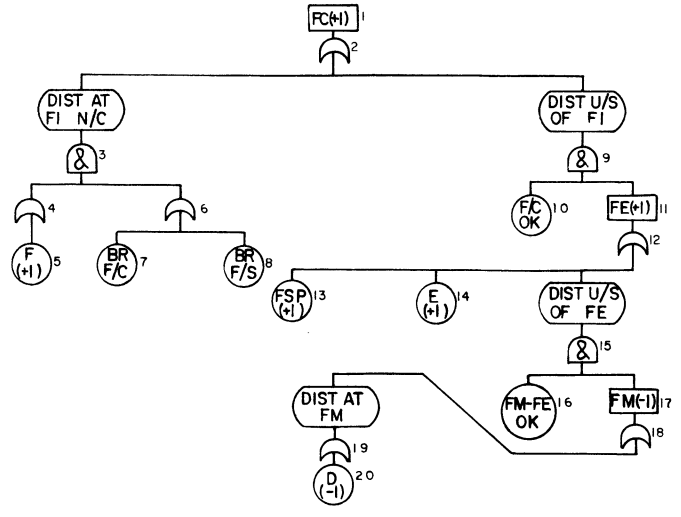


Fig. 4. Flow Control Loop–Original Tree

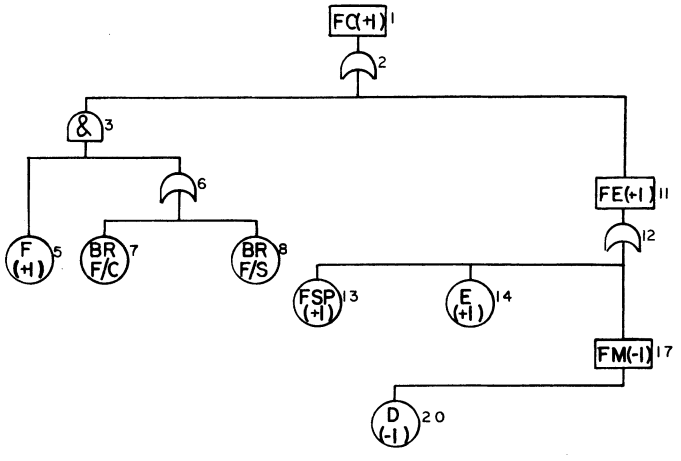


Fig. 5. Flow Control Loop–Reduced Tree

There are 2 points of particular interest in the final tree:

1. The intermediate abnormal condition events FE(+1) and FM(-1) are retained. The modified algorithm generates these events in the original tree. It is important that the algorithm generates them because they might be needed for consistency checks in a complex system. (The retention of these conditions in the final tree is a matter of choice but I prefer it as an aid to understanding the failure mechanisms, with only a small overhead in terms of complexity of representation).

2. The set-point change FSP(+1) appears as a 1-event cut-set. No loop failures have occurred. The event is shown because it demonstrates a mechanism for the propagation of a disturbance (which can be a failure) through the loop, to the variable of interest. The events E(+1) and D(-1) appear as 1-event cut-sets. These events are failures.

V. EXAMPLE #2

Figure 6 shows a more complex digraph constructed by building an outer loop around the one used in Example #1. Figure 6 represents a cascade control-loop commonly found in process plants. The inner loop controls a flow. The outer loop controls a level by means of the flow loop. Single letters are again used to represent external disturbances. I prefer this representation to the use of specific failure events (such as VALVE MECHANISM FAILS OPEN) in order to clarify the mapping from the digraph to the fault tree.

Figure 7 shows the reduced tree for this example. Comparison with Figure 5 shows how the tree generated for the inner loop fits into place. The trees are all laid out such that:

1. Process disturbances are developed on the l.h.s. of the diagram.

2. Set-point and other similar disturbances are developed in the centre.

3. The flow of disturbance information around the loop is developed on the r.h.s.

These layout conventions are for the convenience of the analyst, particularly in regard to ease of assimilation of the final trees.

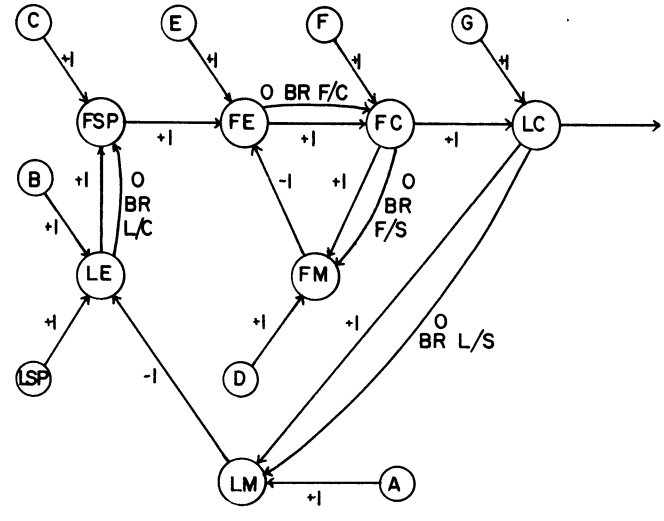


Fig. 6. Cascade Control System Digraph

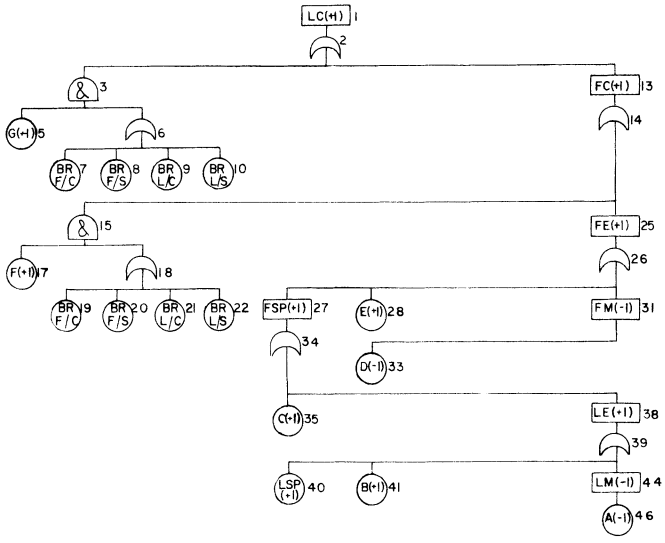


Fig. 7. Cascade Control System – Reduced Tree

VI. THE MODIFIED ALGORITHM

The algorithm used to generate all of these trees is shown in flow-chart form in Figs. 8 and 9. It emphasises the effects of relatively small changes in process variables (i.e. the +1 and -1 states of digraph nodes). The L. & P. algorithm emphasizes two other types of events:

1. Large disturbances (+10 and -10 states of nodes) which the loop cannot compensate. This type of disturbance is effectively handled by the L. & P. algorithm and is not considered further here.

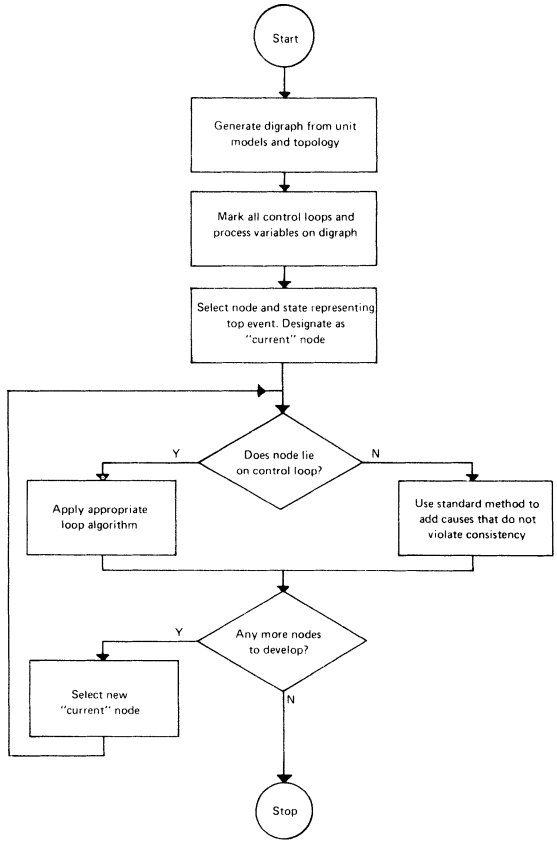


Fig. 8. Basic Fault Tree Synthesis Algorithm

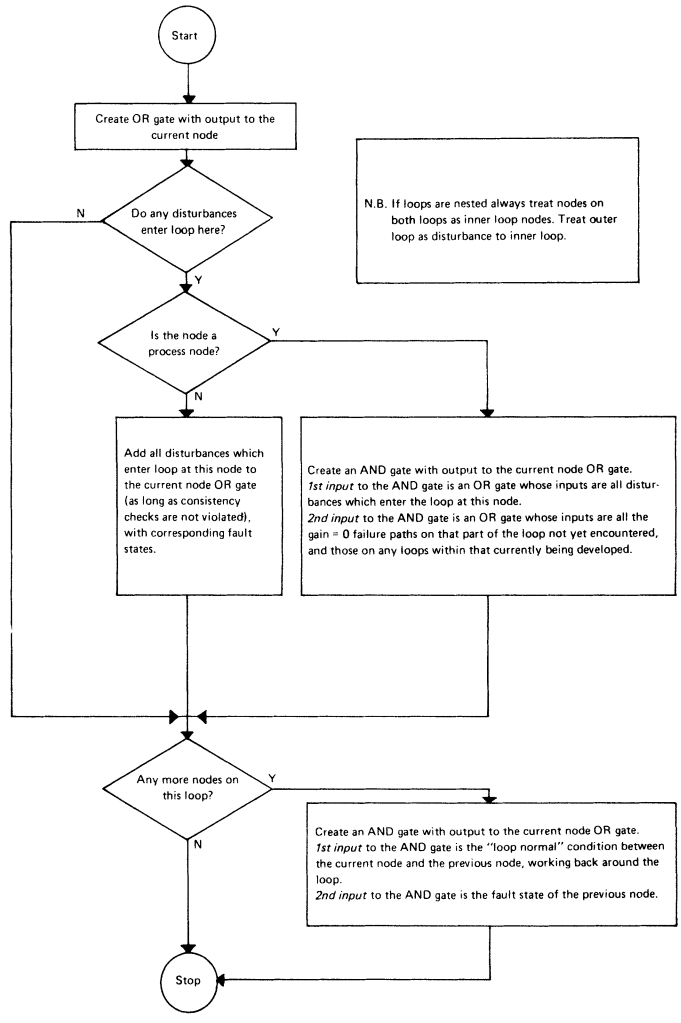


Fig. 9. Algorithm for Development of Node on Feedback Loop

2. Those related to reversed components which cause loop malfunctions. This type is more amenable to treatment using proof-testing at installation or repair time.

The modified algorithm is only intended as a tool to generate the examples for this paper and omits these last two categories of events.

The algorithm is only intended for negative feedback loops since this type of loop is commonly encountered. This paper is primarily oriented to this type of loop.

VII. DISCUSSION

A. *Time-Base Failures*

The apparent simplicity of the example fault trees hides the following implicit assumptions. The modified algorithm includes all zero-gain failure modes on the loop as input to the AND gate used for process node disturbances. This gives cut-sets 1 and 2 in Table 1. These failure modes are easily accepted as causes of the top event FC(+1). The same failure modes appear as cut-sets 5 and 6 in Table 2. Cut-set 5 can be readily visualised as a cause of the top event LC(+1). Cut-set 6 is less obvious. The difficulty arises because of the loop-within-a-loop structure of Example #2. The cases for and against including cut-set 6 are:

Table 1
Cut Sets for FC (+1)

1. F(+1) AND BR F/C
2. F(+1) AND BR F/S
3. FSP(+1)
4. E(+1)
5. D(-1)

Table 2
Cut Sets for LC (+1)

1. G(+1) AND BR F/C
2. G(+1) AND BR F/S
3. G(+1) AND BR L/C
4. G(+1) AND BR L/S
5. F(+1) AND BR F/C
6. F(+1) AND BR F/S
7. F(+1) AND BR F/S
8. F(+1) AND BR L/S
9. E(+1)
10. D(-1)
11. C(+1)
12. LSP(+1)
13. B(+1)
14. A(-1)

1. If the flow sensor fails and the disturbance event F(+1) occurs, then the inner loop fails, FC(+1) will occur and cause LC(+1). The action of the outer loop is irrelevant since if the inner loop were not required then it should be left out of the control scheme altogether. Cut-set 6 is therefore a valid cause of LC(+1).

2. Cut-set 6 describes a failure of the inner loop. When this failure occurs FC(+1) can occur but the outer loop will prevent LC(+1) from occurring. This does not mean that the inner loop is not needed. The inner loop speeds up the control system response to events such as F(+1) in the normal situation where both loops are operative. Cut-set 6 is therefore not a valid cause of LC(+1).

In general it is not possible to resolve these two points of view without further knowledge of the system. The modified algorithm employs the pessimistic rule that the failure mode should be included unless it can be shown to be incorrect. This reinforces the point made in section II that the graph is an imperfect model of the system. If the inner loop controls the flow FC that disturbs the level LC, then there is an implied integration (with respect to time) that is missing from the graphical representation. It would not be particularly useful if it were included in the graph since the graph itself is a halfway stage to the formal fault-tree which is similarly devoid of time-base information.

It may be argued that the philosophy is inconsistent. If time-base information is relevant, then the fault-tree methodology should be extended to reflect this requirement. Unfortunately this extension is non-trivial since much of the theory on which fault-tree manipulation is based is invalidated by introducing a time-base. (For instance the two events X and NOT X are mutually exclusive in the conventional fault-tree. If a time-base is introduced then the two events can occur, in the same branch of the tree, provided that the times of occurrence do not overlap). It can also be argued that it is self-defeating to complicate the tree with a time-base, since one of the primary objects of tree construction is to produce a simple and clear representation of system-fault behavior. Control loops are comparatively sophisticated devices and can exhibit complex transient behavior, particularly when multiple loops interact with one another.

This time-base discussion suggests that there is an inherent mismatch between the problem and the tool being applied. A more complex tool, such as the Cause-Consequence Diagram should be used, if a time-base is important. A more optimistic conclusion is that there are some problems involving control loops which can be treated using fault-trees, but that the results need to be reviewed with great care. The contradiction between these conclusions reinforces the point that a fault-tree, like any model, is an approximation to reality. When a model is used for design purposes (i.e. in a predictive mode) its capabilities are limited by:

1. The correlation between the model and reality.
2. The accuracy required by the prediction.

The major inherent limitations of the conventional fault-tree model are:

1. There is no time-base in the fault-tree.
2. The fault-tree is oriented to discrete-state problems.
3. The fault-tree is most useful when applied to binary-state problems.

In practice many successful analyses have violated one or more of these limitations. All electro-mechanical systems exhibit dynamic responses which violate the time-base limitation and yet most examples in the literature are based on such systems. In these cases the approximation is valid in the context of the problem. If the basic fault-tree methodology is applied to control loops which have complex transients, then the approximations made must be borne in mind, and the results reviewed accordingly.

B. Cut-Set Failures

A further point arises in the 2-event cut-sets shown in Tables 1 and 2. The 2-event cut-sets all assume that:

1. A disturbance occurs.
2. The loop is broken and so cannot respond.

The loop-broken events all have a link-gain of zero in the digraph. This implies that the signal at the output node remains fixed irrespective of changes of the input node signal (Sensor stuck is an example). Consider two cases:

1. The plant is running very steadily. The loop failure occurs. At some later time the disturbance occurs. This sequence is quite correctly represented as a 2-event cut set.

2. The plant is subject to small disturbances affecting the loop. This is normal since the control loop is provided to smooth out such disturbances. After one particular disturbance occurs, the loop responds to maintain the desired value of the controlled variable. At some later time the loop fails. Later still the disturbance dies out (or even changes sign). The loop cannot respond but the controlled variable now moves away from its desired value. At this point only 1 failure-event exists and the 2-event cut set representation is wrong.

This illustrates a weakness of the modified (and original) algorithms. The basic problem is again that time is an important consideration in the scenario given above, viz the time-order of events matters. Algorithms which lead to the type of 2-event cut sets shown in Tables 1 and 2 are inherently

optimistic in that they predict a relatively low-probability 2-event failure mode when a 1-event failure mode might occur.

C. Quality of Computer-Aided Synthesis

Most of the recent work on fault-tree synthesis has been carried out in academic institutions. Significant advances have been made. The L. & P. algorithm (on which most of the comments in this paper are based) potentially covers many real-life problems. I believe that none of the published algorithms, including the modified one used here, is good enough for widespread use by everyone in real applications. The danger with computer-aided synthesis is that the results might not always be checked carefully (this comment applies to all computer-aided syntheses, not just to fault-tree synthesis). Computer-aided fault-tree synthesis will fall into disrepute if it is widely applied before the technology is fully developed.

The previous statement is particularly true of any aid used in safety and reliability analyses. There is also a philosophical difficulty when computer-aided synthesis methods are applied to real plants. Consider two extreme viewpoints:

1. If unproven techniques are applied then the results have to be checked so carefully that computer-aided synthesis is best considered as an independent check.

2. If proven techniques are applied so that we are certain that high-quality trees are produced then, in practice, less human effort will be expended on safety studies. This is counterproductive because the real value of drawing fault-trees was that the analyst gained a good insight into system behavior.

I believe that there is a place for computer-aided synthesis between these two extremes. As a design evolves a skilled analyst can use computer-generated trees to find high-probability failure modes. These failure modes can be eliminated by design changes and new trees generated quickly, as a check. This style of use still involves considerable human interaction but also requires that the analyst be confident of high quality computer-aided analyses.

I believe that the published algorithms are not yet of sufficiently high quality for general use. Criteria must be established so that algorithms can be evaluated. Algorithms must be examined independently, debated at a detailed level and test-cases applied. This paper contributes to the validation debate.

ACKNOWLEDGMENTS

I am grateful for the following support:

1. From the Science Research Council for the related project "Alarm Analysis using a Process Computer."

2. From NATO for allowing me to attend the 1978 Advanced Study Institute "Synthesis and Analysis Methods for Safety and Reliability Studies."

3. From Professor D.C. Freshwater and Professor F.P. Lees for providing research facilities within the Department of Chemical Engineering at Loughborough University of Technology.

References

[1] J.B. Fussell, "Synthetic tree model–a formal methodology for fault-tree construction," Aerojet Nuclear Report ANCR–1098, 1973 March. Available from National Technical Information Service; U.S. Dept. of Commerce; 5285 Port Royal Road; Springfield, Virginia 22151 U.S.A.

[2] G.J. Powers, F.C. Tompkins, "Fault-tree synthesis for chemical processes," *AICHE Journal*, vol 20, 1974 Mar, pp 376-387.

[3] P.K. Andow, F.P. Lees, "Process Computer Alarm Analysis: Outline of a Method Based on list Processing," *Trans I. Chem. E.*, vol 53, 1975 Oct, pp 195-208.

[4] G.A. Martin-Solis, P.K. Andow, F.P. Lees, "An approach to fault tree synthesis for process plants," in "Proc 2nd International Symposium on Loss Prevention and Safety Promotion in the Process Industries," Heidelberg 1978. Available from: Dechema; Deutsche Gesellschaft fur chemishces apparatewesen; Frankfurt (MAIN), Fed. Rep. Germany.

[5] G.E. Apostolakis, S.L. Salem, J.S. Wu, "CAT: A Computer Code for the Automated Construction of Fault Trees," Report No. EPRI-705, 1978. Available from: Electric Power Research Institute; 3412 Hillview Avenue; Palo Alto, California 94304 U.S.A.

[6] S.A. Lapp, G.J. Powers, "Computer-aided synthesis of fault-trees," *IEEE Trans. Reliability*, vol R-26, 1977 Apr, pp 2-13.

[7] J.R. Taylor, E. Hollo, "Algorithm and programs for consequence diagram and fault tree construction," Report No. Riso-M-1907, 1977. Available from: Library of the Danish Atomic Energy Commission; Riso; DK-4000 Roskilde, Denmark.

[8] E.J. Henley, H. Kumamoto, "Comment on: Computer-aided synthesis of fault trees," *IEEE Trans. Reliability*, vol R-26, 1977 Dec, pp 316-317.

[9] M.O. Locks, "Synthesis of fault trees: An example of non-coherence," *IEEE Trans. Reliability*, vol R-28, 1979, Apr, pp 2-5.

[10] H.E. Lambert, "Comment on the Lapp-Powers Computer-aided synthesis of fault trees," *IEEE Trans. Reliability*, vol R-28, 1979, Apr, pp 6-7.

[11] T.W. Yellman, "Comment on: Comment on computer-aided synthesis of fault trees," *IEEE Trans. Reliability*, vol R-28, 1979, Apr, pp 10-11.

[12] S.A. Lapp, G.J. Powers, "Update of Lapp-Powers fault-tree synthesis algorithm," *IEEE Trans. Reliability*, vol R-28, 1979, Apr, pp 12-15.

KEY FOR FIGURES

Links:

Solid arrows are links defined within model
Dashed arrows are links defined by adjacent models

Link gains:

0	No interaction
1	Moderate change in cause node gives moderate change in effect node
10	Moderate change in cause node gives large change in effect node
+	Change in same direction
-	Change in opposite direction

Node names:

M1, M2	Mass Flows
P1, P2	Pressures

T1, T2	Temperatures
S3	Control signal
FSP, LSP	Set Points
FE, LE	Errors
FM, LM	Measurements
FC, LC	Controlled Variables
A, B. . . . . .G	Disturbances

Abbreviations:

BR	Broken
RV	Reversed
DIST	Disturbance
U/S	Upstream
N/C	Not Controlled
F/C	Flow Controller
F/S	Flow Sensor
L/C	Level Controller
L/S	Level Sensor

BIOGRAPHY

Dr. Peter Andow; Chemical Engineering Department; University of Technology; Loughborough; Leicestershire LE11 3TU; United Kingdom.

Peter Andow is a Lecturer in Chemical Engineering at the University of Technology, Loughborough. He is studying fault propagation in process plants for design and real-time purposes. He received a B.Tech from Loughborough, MChE from Delaware and PhD from Loughborough. He has worked as an engineer with Unilever on food and detergent products and as a system safety engineer with British Nuclear Design and Construction.

Manuscript TR78-149 received 1978 November 13; revised 1979 May 12, 1979 July 17. ☆☆☆

Book Review

Ralph A. Evans, Product Assurance Consultant

MDR-12, Digital Failure Rate Data
David B. Nicholls, 1979, $50.00 (nonUSA $60), 412 pp.
Reliability Analysis Center; RADC/RBRAC;
Griffiss AFB, NY 13441 USA.

Table of Contents

1. Digital Summarized Data 140 pp
Introduction
Summarized Generic Failure Rates - Field Data
Digital Microcircuit Observed and MIL-HDBK-217B Predicted Failure Rates
Summarized Generic Replacement Rates - Field Data
Summarized Generic Failure Rates - Reliability Demonstration and Equipment Checkout Data
Summarized Generic Failure Rates - Life Test Data
2. Digital Device Data - Detailed Listings 266 pp
Introduction and Usage Guide
CMOS, PMOS, DTL, ECL, TTL high speed (HTTL), TTL low power (LTTL), TTL Schottky (STTL), TTL low power Schottky (LSTTL), TTL SUHL, TTL

This report is apparently an updating of MDR-8 with the same title from 1978, although this fact is not stated explicitly. This report, and this kind of report in general, are good in that they summarize many data that would otherwise not be accessible to many engineers. They are capable of gross misuse by those who interpret the numbers as some kind of truth table for the future. The quality and reliability of purchased devices is affected by a tremendous number of variables which are beyond the control of the user; screening and testing can help of course, but there's at least a little blind luck involved from the user's point of view.

The user of this report does not know many pertinent facts about the failed items, especially as contrasted with the devices he will be using. But the data and summaries in this report can give a rough idea of what to expect and what to watch out for. Even those companies who keep great amounts of their own data can profit from comparing their data with this report. The comparisons of "predicted by MIL-STD-217B" with several kinds of experience are especially helpful.

Many of the categories have very few failures (e.g., less than 10). That means the statistical uncertainty is very great. The report uses a symmetric 60 percent statistical confidence interval as the measure of uncertainty. That is a low value of statistical confidence, but is used (unwisely, I think) to keep the interval reasonably narrow. Generally the instructions and cautions in the report are good and honest, but they do not lean over backward to browbeat those who have too much blind adoration for the printed word.

MDR-13, Memory/LSI Data
Mark R. Klein, 1979, $50.00 (nonUSA $60.00), 209 pp.
Reliability Analysis Center; RADC/RBRAC;
Griffiss AFB, NY 13441 USA.

Table of Contents

Introduction 2 pp
Definitions of Terms 2 pp
1. Memory/LSI Summarized Data 46 pp
Introduction
Summarized Generic Failure Rates - Field Data
Memory/LSI Observed and MIL-HDBK-217C Predicted Failure Rates
Summarized Generic Failure Rates - Reliability Demonstration and Equipment Checkout Data
Summarized Generic Failure Rates - Life Test Data
2. Memory/LSI Data - Detailed Listings 158 pp
Introduction and Usage Guide
Random Access Memories (RAM)
Read Only Memories (ROM)
Programmable Read Only Memories (PROM)
Shift Registers
Microprocessors
LSI-MSI Complex Standards

This is the same kind of report as MDR-12 reviewed above on this page. The review for that report holds for this one except that these devices are even more complicated (in terms of processing, layout, and number of elements). Therefore the cautions in the above review should be taken to heart even more strongly. * * *