PLOTTING AND ANALYSIS OF FAULT TREES IN SAFETY EVALUATION OF NUCLEAR POWER PLANTS

by

A. Wild

Atomic Energy Control Board
Ottawa, Canada

REPORT

December 21, 1979
CONTENT

Introduction.......................... 1
Fault Tree Method.................. 3
Fault Tree Description File........ 12
Fault Tree Plotting.................. 15
Logically Identical Structures..... 23
Logical Analysis...................... 40
Numerical Evaluation............... 43
Principle of Operation.............. 48
Software Package.................... 52
Appendix 1.......................... 64
Appendix 2.......................... 67
INTRODUCTION

A modern nuclear power plant has hundreds of thousands of parts ranging from huge concrete structures to tiny switches. In spite of its complexity a modern reactor is reliable and safe.

One important step in achieving the high reliability and safety is an extensive safety analysis. The safety analysis is not limited to major causes of spectacular accidents such as a steam pipe break, but includes also checks of many other causes, including burnt-out fuses and errors in computer programs. As a general rule, any component is considered a potential safety hazard - unless proven otherwise.

This poses a huge problem for the safety analyst in the design organization who has to check every item. The large amount of information involved also poses a problem for the licensing authority which has to verify that the safety analysis is complete, that none of the potential safety hazards was erroneously excluded, and that the expected risk of radiation releases does not exceed the values defined by stringent regulations. To achieve this, many different tools and methods are used, one of which is the fault tree (ref. 1).
The fault tree indicates graphically how a failure of a system or equipment can be caused by failures of its parts and by other events (e.g. operator errors). The main strength of the fault tree method is the ability to detect cross-links between systems - such as a failed power supply simultaneously disturbing reactor control, and disabling the safety equipment protecting against such a disturbance. However, this strength can be used fully only if fault trees are constructed for complete nuclear generating stations and are not limited to only parts of systems. Such trees are inevitably large and have to be handled by computers. Many computer programs for fault trees analysis exist (ref. 2, 3), but they are designed for use on big computers and mostly with cumbersome inputs and difficult to understand outputs (ref. 4). This does not represent a severe limitation for the use of fault trees by reliability specialists, but does form a barrier for general acceptance of the fault tree method as an engineering tool.

In an attempt to promote the use of fault trees, AECB's Safety Evaluation Division has developed a new fault tree handling system for small computers.

The most outstanding characteristics of the system are: simple free format input, graphical output, and evaluation of fault tree logic by a reduction algorithm based on pattern recognition. Although the system is intended primarily for examining the fault tree logic, it is also capable of numerical evaluation of probabilities, steady state unavailabilities and failure rates. The system includes extensive error checking of both fault tree logic and numerical data.
The system is presently running at AECL on a HP-1000 computer (16 bits) with disc drive, graphics terminal and X-Y desk top plotter. It can be accessed remotely from a CRT or printing terminal with 300 baud modem. If the fault tree is too big for the desk top plotter, it can be either plotted in segments or the system can produce a plot control file which is sent through a 2000 baud modem to a larger computer installation with a drum plotter.

When the software for the system was developed, the design intent was not only development of a system for internal use by AECL staff, but also development of software which could be used by other users within the nuclear industry.

This report describes the system. According to available information, it is the first description of a complete fault tree processing system for a small computer. The systematics of fault tree patterns is also published here for the first time.

**FAULT TREE METHOD**

A fault tree is a logic diagram indicating how the failure of a system or equipment can be caused by failures of its parts and by other events (e.g. operator errors) either individually or in combination.

A fault tree is constructed from the top down. The procedure starts with a definition of the "top" event (e.g. radiation release or system failure) and identifies in a systematic way its causes (the originating faults or events).
The procedure continues until the basic faults or events are reached, the characteristics (e.g. frequency of occurrence) of which are known either from observations or from physical properties. It is an opposite approach to the Failure Mode and Effect Analysis, where a single component failure is postulated first and the possible results determined afterwards.

The basic building blocks are the AND-gates and the OR-gates. An OR-gate is used if any of the input faults (events) can cause the output fault (event) - either individually or in combination with other input events. An AND-gate is used if all input faults (events) have to occur simultaneously for the output event to occur. Although it would be possible to construct quite large trees with AND's and OR's only, some other symbols are also used to enhance the clarity of the tree. The symbols are shown on Fig. 1.

An m-out-of-n redundant structure is identified by a square with a number defining the type of redundancy. In a 24 (two-out-of-four) structure four identical units are used of which at least two have to be operational. Such a system fails if at least three units fail (4-2+1). The square symbol replaces in this case a grouping of one OR-gate and four AND-gates.

A WHILST-gate is a two-input AND-gate where one input defines an event (e.g. process failure) and the other input a condition (e.g. safety system unavailable) which has to exist at the time of the first event for the output event (e.g. accident) to occur. A distinction between a WHILST-gate and a AND-gate is required for numerical evaluation only.
FAULT TREE SYMBOLS

- OR-gate
- AND-gate
- WHILST-gate
- REDUNDANT STRUCTURE (e.g., 2 out of 3)
- EXCLUSIVE-OR-gate
- NOT (INVERSE OF CONDITION)
- BASIC EVENT
- UNDEVELOPED EVENT
- ANALYSED ELSEWHERE
- BASIC CONDITION (OPTIONAL SYMBOL)
- CONDITION ANALYSED ELSEWHERE (OPTIONAL)
- TRANSFER-OUT (EVENT ALSO APPEARS ELSEWHERE)
- TRANSFER-IN (COMMON EVENT)
- COMMON EVENT, TRANSFERRED FROM ANOTHER SHEET
- HOUSE, EVENT WHICH ALREADY HAPPENED

FIGURE 1
An Exclusive-OR-gate is used for identification of a situation in which any one of the input events can cause the output event to occur, but if all input events occur simultaneously, the output event does not occur. An Exclusive-OR-gate defines, for example, damage of a transistor in a amplifier with two balanced voltage suppliers. If the negative voltage supply fails alone, the remaining positive voltage supply will cause the transistor to burn-out. If the positive voltage supply fails alone, the remaining negative voltage supply will cause a break-down. If both voltage supplies fail simultaneously, the transistor will not be damaged. (The amplifier will not work, but that is a different event). An Exclusive-OR situation has to be properly identified and carefully analysed, particularly in cases where the designer attempts to counteract one event (e.g. failure of the positive voltage supply) by forcing the other event (e.g. switching OFF the negative voltage supply), because improper timing may make such a protection ineffective.

Basic events are identified by circles. Events which by their nature can not be considered basic, but for which the logic is not further developed are identified by DIAMONDS. A circle within the DIAMOND indicates that the event is analysed elsewhere.

An event which has already occurred is identified by a HOUSE. This symbol is used in fault trees for analysis of degraded operations, e.g. during certain maintenance procedures or repairs.

Common events used as inputs to several gates in the tree are identified by
triangles. In a group of identical events one event is marked as transfer-out, the remaining events are marked as transfer-in.

If a tree uses WHILST-gates, it is desirable to make a distinction between events (failures) and conditions (e.g. unavailable systems, atmospheric conditions, presence of gases etc.). Basic conditions are identified by omitting the basic event circles. Should this omission lead to some graphical or logical difficulties, an oval is used to replace the circle.

The symbols used in the fault tree are derived from symbols used in diagrams for electronic logic circuits. In the same way as electronic logic is binary, the fault tree is also binary. A fault either does exist (logical one) or does not exist (logical zero). Some methods for fault trees with multilevel logic have been developed, but at this stage they seem to be only of theoretical interest (ref. 5).

The fault trees are systematic records of logical statements of various ways by which a system can fail to perform a defined function. They are thus basically qualitative, but may also provide a suitable framework for quantitative analyses.

One great advantage of the fault tree in comparison with other methods is the possibility of expanding the fault tree as the knowledge of the system increases, without having to rework what has already been done. The expansion can proceed by both a further development of "basic" events and merging of trees
(e.g. with top events corresponding to failures of individual systems) into one "super-tree" (e.g. with "radioactivity release to the public" as a top event). This capability of merging trees from partial analyses makes the fault tree method a powerful tool for analysing a complex system with cross-links between parts of the system.

Fig. 2 shows an example of a fault tree for one channel of a four-channel computer controlled reactor trip with computers communicating through a data link (ref. 6). The fault tree was plotted by the AECB computer from the information in the fault tree description file on Fig. 3. The format of the plot corresponds to that used in ANSI N41.4-1976 (ref. 7) with the fault tree developed horizontally and the "top" event on the left.

The fault tree on Fig. 2 is an example of a safety system, where a complex linking of computers was incorporated for economical reasons (protection against spurious trips) rather than to improve the reliability of the safety function (reactor trip from channel A). The example shows the importance of analysis of complete systems. Without a complete fault tree analysis it would be easy to consider the data links between computers as "non-safety-affecting equipment" and connect them to a common power supply. Logical reduction of the fault tree on Fig. 4 as produced by the computer shows that a simultaneous failure of all computer links (installed "just for economic reasons") would completely disable the reactor trip. However, to detect this effect, the fault tree has to be sufficiently developed, which means also large - and manageable only with the aid of a computer.
FIGURE 2 - Fault Tree for Computer Controlled Reactor Trip
FIGURE 3 - Fault Tree Description

File
FIGURE 4 - Rearranged Fault Tree from Figure 2
FAULT TREE DESCRIPTION FILE

One of the keys to success of any computer system is an easy input of data. Preparation of data should not require special forms, it should not be necessary to count columns, and the errors - at least the more common ones - should be detectable. In the HP-1000 computer all input data is stored in a disk file. Because all processing is controlled directly by commands typed on the terminal, no directives (job control cards) are required. However, some other computers used for fault tree analysis may require directives. The design of the fault tree description file therefore allows the use of directives, so that files can be exchanged between computers. If a file prepared for a computer requiring directives is processed on the HP-1000 (or other interactive computer), the directives are simply skipped.

The fault tree description file, which contains all the information for plotting and analysing the fault tree, has the following structure:

(Leading Directives)

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATE DESCRIPTIONS</td>
</tr>
<tr>
<td>Event names</td>
</tr>
<tr>
<td>DATA SEPARATOR</td>
</tr>
<tr>
<td>BASIC EVENT DESCRIPTIONS</td>
</tr>
<tr>
<td>Event names</td>
</tr>
<tr>
<td>(Trailing directives)</td>
</tr>
</tbody>
</table>

- gate description block
- optional basic event description block
Information indicated by lower case letters and the whole basic event description block are optional. Use of leading and trailing directives depends on the requirements of the computer. (Not required for HP-1000).

The sequence of structural blocks must be maintained. However, no sequencing of records (lines in the file) within a block is necessary, e.g. the top gate description may be the last record in the gate description block.

Records with event names are not limited to a particular block, e.g. the name of a gate may be defined in the basic event description block and vice versa.

Records with comments (asterisks in the first column) may be anywhere in the file. Events are identified by codes, formed by a group of up to ten characters.

The title, if used, must be the first record in the file (after leading directives and comments which are skipped by the program). The title has the format of a name record. It begins with a title code, followed by a space, an apostrophe, and up to 40 characters of title. The title code is used by the program only for format identification and it may be just a dummy code.

The gate description starts with the event code, followed by a space and the gate identification group. The first character of this group determines the gate type, the second character is a number (1 to 6) which defines the number of inputs.

- An AND-gate with n inputs
- Rn OR-gate with n inputs
- W2 WHILST (second input is the condition)
- mn m-out-of-n redundant structure
Xn  EXCLUSIVE OR gate
Nl  NOT (for conditions only).

The codes of input events follow the gate identification group. They are
separated by spaces.

The number of spaces used as separators is optional. More than one space
can be used.

The event name is defined by a record which starts with the event code
followed after at least one space by an apostrophe (') and up to forty
characters of event name which is split on plots in two lines, each twenty
characters long. Should the first line be shorter than twenty characters, it
can be terminated by a semi-colon (;) followed immediately by the second line.

Basic events can be defined in an optional block which follows the data
separator formed by a single word DATA starting in the first column.

Basic events are displayed as circles. They can be changed by basic event
description "cards" which start with an event code followed by a space and one
letter event identifier.

H - event which has already happened (house)
D - undeveloped combined event (diamond)
E - event analysed elsewhere (diamond with circle)
In fault tree description files used for numerical evaluation the basic event descriptions are compulsory and must contain numerical data, identified by an octothorp (#). The system accepts several formats of data, depending on the type of data. Probabilities are expressed by numbers (e.g. 0.002 or 2E-3). Failure rates are expressed in failures per year, month, day or hour (e.g. 3.5/Y, .1/M, 2.5E-5/H.) They may be also expressed in BITTS (10^-6 per hour, e.g. 25B) and FITS (F, 10^-9 per hour), or indirectly by Mean Time Between Failures (e.g. 120Y). Unavailabilities can be expressed either by a single number, or by failure rate (or MTBF) as above and Mean Time To Repair (e.g. 25Y 7H or .4/Y 7H). Reliability (probability of failure within a time period) is expressed by failure rate (or MTBF) and the time period (e.g. 120Y FOR 3.5H).

Further details of the file are described in Ref. 8.

FAULT TREE PLOTTING

The basic fault tree handling procedure is plotting. The terminology used for describing the fault tree implies a vertical orientation of the tree, with the "top" event on the top. This orientation is well suited for examples in textbooks and technical papers where events are identified by numbers or short codes. For fault trees used in actual analyses, when the codes and especially names (descriptions) of events are long, the horizontal orientation with the "top" event on the left leads to a more compact presentation. However, as we are used to read from left to right and from top to the bottom, the change of orientation is not just a rotation but rather a restructuring where "top" and
"left" interchange their positions, as can be seen from comparison of a plot on Fig. 2 with its description file on Fig. 3.

The coordinates of events are calculated by the program. The variety of available symbols is on Fig. 5. Some of the symbols have a restricted use. The star indicating an error and the flags indicating common events are inserted only by the program and cannot be input from the fault tree description file. In fault trees used for logical and numerical evaluation further restrictions apply to the use of some symbols and also to the use of some structures. The only structural restriction for plotting applies to the WHILST-gate which must not be used in the condition branch of another WHILST.

In order to fulfill various requirements for plotting, two different formats are available. The first format identifies all events with codes and full names. The second format identifies events by codes only as shown on Fig. 6. When the second format is used, the codes may be optionally replaced by short names (up to ten characters).

The easiest way to control plotting is from a graphics terminal. However, the small size and limited resolution of the screen prevents full display of the whole fault tree. Therefore, a graphics display control program makes it possible to scan the tree in sections and to display a simplified skeleton of
FIGURE 5: - Variety of Symbols Used for Plotting - Mod. 1
the tree. The control program also provides a limited tree editing capability. This includes: squeezing (elimination of lower level events as used on gates in the lower right corner of Fig. 2 and Fig. 4), switching (re-arranging inputs for a gate), cutting-out (suppressing display of inputs to a gate) and insertion of extension lines (to decrease the plot width). A computer routine was also developed for elimination of gates - if such an elimination is logically permissible. All fault tree editing is limited to structural modifications not affecting the fault tree logic as defined in the fault tree description file. Requests for modifications violating the fault tree logic are rejected by the program.

When the plotting is done locally on the desk top plotter, proper positioning of the plot, or selection of the desired segment, is achieved by a "window". The size of the "window" depends on the size of paper used.

When plotting on a remote drum plotter is required, the computer program produces a plot control file with all the information required for plotting. An example of the plot control file is on Fig. 7.

The plot control file can be also created by commands from a terminal without graphics capability. A fault tree printing routine allows operators of these terminals to verify fault tree structure before the plot control file is created. The print is a vertically oriented skeleton of the tree with event codes truncated to the last six characters, as shown on fig. 8.
This format was selected to provide optimum legibility of the information, given constraints posed by the printers. For full verification of the plotting information, including the event names, the printed skeleton may be used in conjunction with a print-out of the plot control file. The task of comparing the two sources of information is simplified by sequencing of lines in the plot control file which follow the fault tree skeleton structure as can be seen from comparison of Fig. 7 and 8.

The width of the printed skeleton is limited to 72 columns to allow printing even on small portable terminals. The limited width also allows to include the printed skeleton directly in reports without any reduction of size, as mentioned in the section on numerical evaluations. Skeletons of large trees are broken down in overlapping sections with alignment marks.

The fault tree plotting routine is called on the HP-1000 computer by commands TR,FTP or TR,FTB.

When the graphics terminal HP2678A at the AECB is used, the plotting routine is called by the command TR,FTP. At the time of writing this report, the available memory was only 18 k-words which limited the size of the displayed and plotted tree to 120 events. Larger fault trees with up to 480 events can be displayed and plotted in segments by the command TR,FTB. The routine FTB is used also for smaller trees, when it is desirable to plot separately just one branch of the tree. It should be noted that in this system plotting of a fault
PLTCNTRLFL 6 252 REACTOR TRIP CHANNEL A
TRCHA 11 68 12 123R NO TRIP FROM CHANNEL A
TRDAM 21 123 36 210A NO TRIP DEMAND
RTBA 21 12 12 12D TRIP BREAKER A FAILED
TRLLA 31 210 180 240R NO LOCAL TRIP
TRGLA 31 36 0 72R NO GLOBAL TRIP
TRLCS 41 240 228 252R NO TRIP SIGNAL
TRLEA 41 180 156 204A NO TRIP ENABLE
TRDGL 41 72 24 12023
TLCA 42 0 0 0< TRIP LOGIC COMPUTER A FAILED
CHNA 51 252 252 252E TRIP FUNCTION SENSING CHANNEL A
TLCA 54 228 228 228O TRIP LOGIC COMPUTER A FAILED
TRPLBA 52 204 204 204< NO PARTIAL TRIP RECEIVED FROM B
TRPLCA 52 180 180 180< NO PARTIAL TRIP RECEIVED FROM C
TRPLDA 52 156 156 156< NO PARTIAL TRIP RECEIVED FROM D
TRPLBA 54 120 108 132R NO PARTIAL TRIP RECEIVED FROM B
TRPLCA 54 72 60 84R NO PARTIAL TRIP RECEIVED FROM C
TRPLDA 54 24 12 36R NO PARTIAL TRIP RECEIVED FROM D
TRDLBA 61 132 132 132E DATA LINK FAILURE B TO A
TLCB 61 108 108 108O TRIP LOGIC COMPUTER B FAILED
TRDLCA 61 84 84 84E DATA LINK FAILURE C TO A
TLCB 61 60 60 60O TRIP LOGIC COMPUTER C FAILED
TRDLDA 61 36 36 36E DATA LINK FAILURE D TO A
TLCB 61 12 12 120 TRIP LOGIC COMPUTER D FAILED

FIGURE 7
Plot Control File
Figure 8: - Printed Skeleton
tree in segments does not result in loss of proper identification of common events.

Non-graphics terminals such as VUCOM or MINITERM can create plot control files using the command TR,FTL. This initiates the logical analysis program containing a sub-routine for creating a plot control file.

A description of the use of routines FTP, FTB and FTL is given in ref. (9).

LOGICALLY IDENTICAL STRUCTURES

The structure of a fault tree is not unique. A top event may be defined by several fault trees with different structures, which are logically identical and lead to the same analytical results. A simple example of two logically identical fault trees is on Fig. 9.

The existence of logically identical structures is exploited in several ways. It allows the analyst to record his thoughts without worrying about rigid rules. If the relation of events is not fully clear, it may be helpful to construct for the same top event several fault trees using different logical approaches and compare the results. Re-structuring the tree may also be of considerable help in utilizing data for numerical analysis. It often happens that data for a particular event is not available, but data for a certain grouping of events is known from operating experience. Re-structuring provides a means for using the available data. Several other possibilities for use of restructuring exist. Perhaps the most important for the purpose of this report
FIGURE 9 - Two Logically Identical Trees
is its use in reduction of common events. E.g. if two pumps used in parallel are supplied from the same water source, unavailability of water will disable both pumps. The fault tree on Fig. 10 with a common event can be re-structured in a logically identical fault tree on Fig. 11, where unavailability of water became an independent event, which is analytically much easier to handle.

The re-structuring process used in reduction of common events consists of a repeated application of seven steps, which can be called shift, unification, replacement, switch, masking, erasure, and jump. The steps are based on rules of Boolean algebra. Each step has two complementary forms, in which the AND and OR gates are exchanged.

It should be noted that re-structuring maintains the logical identity of the top event. However, logical meaning of other events defined by gates in the tree may be changed in the process. The affected events are marked by asterisks - both in the examples and in actual use of the program.

**Shift**

In a chain of OR's any event can be moved from one level to another level. The shift may span several levels as long as the chain of OR's is not interrupted by a gate of any other type. Shifting is also possible in a chain of AND's. Examples of shifting are on fig. 12. The event subjected to shift is marked by XXXX. This highlighting of events directly involved in the particular restructuring step is used also in all subsequent examples.
FIGURE 10
Fault Tree for Two Pumps in Parallel

FIGURE 11
Restructured Fault Tree from Fig. 10
FIGURE 12

Shift
Unification

Two or more identical events connected to the same gate can be replaced by a single event. It seems trivial, but it is this step which leads to a reduction of common events. In most cases the unification step immediately follows a shift or switch and forms with them virtually a single step (implied unification). An example of common event reduction using shift and unification is on Fig. 13.

Replacement

If an AND or OR gate has only one single event, the output event of that gate can be replaced by the input event. Fig. 14 shows an example of replacement used in conjunction with shift and unification.

Switch

A structure of OR-of-AND's with the OR gate on the higher level can be replaced by a logically equivalent structure of AND-of-OR's with the AND gate on the higher level - and vice versa. The switch - as used in fault tree reduction - always involves a structure with common events. A basic example of a switch is on Fig. 15. It is obvious that the statement "top event occurs if either events C and D occur together, or events C and E occur together" which describes the first structure of Fig. 15 has the same meaning as the statement "top event occurs if event C occurs together with either D or E" which describes the switched structure on Fig. 15. Example of a switch applied to an AND-of-OR's structure is on Fig. 16.

Switch can be applied also to structures with gates with more than two
Figure 13
Unification

Figure 14
Replacement

Figure 15
Switch in Or-of-AND's
inputs as shown on Fig. 17. If only some of the lower level gates contain common events, a preliminary step involving shift is required - Fig. 18.

Re-structuring on Fig. 17 and 18 required the computer to insert new artificial events. The codes of those events are selected in such a way that no confusion is possible with other event codes. The codes use a series of '#'s with imbedded blanks which would disqualify these codes for use in fault tree description files.

A particular case of the OR-of-AND's structure is the OR-of-WHILST's structure, because WHILST is essentially an AND gate with two inputs belonging to different categories (failure and unavailability). Examples of OR-of-WHILST's are on Fig. 19.

**Masking**

The statement "T occurs if either C occurs alone, or C occurs together with D" can be replaced by a shorter "T occurs when C occurs". Fault tree depiction of those statements is on Fig. 20. The event C masks event B consisting of C and D. A similar masking in a structure with AND gate on top is on Fig. 21. Fig. 22 shows the masking event imbedded in a string of OR gates.

Masking can also occur in structures containing longer strings of gates. The condition for masking a gate can be determined by the following Unification Algorithm for Masking. For a common event find the leading event in a string of OR(AND) gates. This event becomes the top event of a branch where masking can occur. A gate in the branch is masked if it is an AND(OR) gate containing the
Figure 16
Switch in AND-of-OR's

Figure 17
Switch with multi-input gates
Figure 18
Switch with insertion of buffer gate

Figure 19
Switch in OR-of-WHILST's
FIGURE 20
Masking in OR-AND Structure

FIGURE 21
Masking in AND-OR structure

FIGURE 22
Imbedded Masking
common event, and the path from the gate to the top event of the branch is not interrupted by a gate which is used also in another branch, or a NOT gate, or an Exclusive-OR gate.

Erasure

Erasure can occur in a three-level alternate chain of AND and OR's. It is similar to masking but eliminates only an input to a gate instead of a whole gate. Examples of erasure are on Fig. 23 and 24. The erasing event can be imbedded as shown on Fig. 25.

Erasure can also occur in structures containing longer strings of gates. The condition for erasure can be determined by the Unification Algorithm for Erasure which is similar to the unification algorithm for masking, but both the masking event and the masked event must in the same type of gate (AND or OR). A corollary of this algorithm applies also to m-out-of-n redundant structures.

Erasure in a m-out-of-n redundancy leads to a decrease of n by one, or whatever the number of erasing events is. The example on Fig. 26 shows a redundancy degradation from 2-out-of-4 to 2-out-of-3. If n decreases so that it equals m, the redundant structure degrades into a single OR-gate.

Erasure and also the Jump described in the next paragraph are difficult to explain in simple terms. The validity of these steps is best demonstrated by a comparison of the minimal cut sets for the trees before and after the step.
FIGURE 23
Erasure in AND-OR-AND

FIGURE 24
Erasure in OR-AND-OR
FIGURE 25
Imbedded erasing event

FIGURE 26
Erasure in redundant configuration
Jump

A jump can occur in a four-level alternate chain of AND and OR's with a double occurrence of a common event. One of the events must form a "stepping stone". Examples of jumps are on Fig. 27 and 28. The "stepping stone" event may be imbedded - Fig. 29. A special case of a "stepping stone" imbedded in a redundant m-out-n (23) structure is on Fig. 2, where "the stepping stone" must consist of at least m (in this example two) events of the jumping AND-gate.

(see event TRLEA on Fig. 2 and Fig. 4).

Conditions for a jump are quite restrictive. The gates of the chain in the branch to the stepping stone must conform to the patterns shown and the third gate in the chain (counted from the bottom common event) must have only two inputs. Although a jump does not result directly in any reduction of the fault tree, it is often a key to a solution of an otherwise irreducible structure.

Inversed Events

An event derived from another event through a NOT-gate forms its inverse. Although an event and its inverse are two different events, their characteristics are so tightly related, that they form a doublet resembling common events subjected to unification and erasure.

If an event and its inverse form inputs to a gate, the output of the gate becomes an impossible event if the gate is an AND. The output becomes a "HOUSE" condition (which always exists) if the gate is an OR. Several examples of unification of inversed events are on Fig. 30. Some of the events are imbedded in chains of identical gates. In a alternate AND-OR or OR-AND structure inversed events cause erasure shown on Fig. 31.
FIGURE 27
Jump using "stepping stone" in OR

FIGURE 28
Jump using "stepping stone" in AND
FIGURE 29
Jump using imbedded stepping stones

FIGURE 30
Unification of inversed events
Identification of Identical Gates

In practical application of fault trees it may happen that two gates of the same type and with identical inputs are assigned different identification codes. This may be caused by an inadequacy of the rules for coding events, which define assignment of codes for basic events (e.g. component failures) but leave considerable freedom in assigning codes for composite events expressed by gates. Also some originally different gates may become identical during the structural changes associated with the seven steps described above. Finally, for some structures with multiple input gates it is not obvious that identical gates are imbedded. An example of such a structure is on Fig. 32. Improper identification of identical gates interferes seriously with the steps defined above. The program designed for carrying out these steps has therefore built-in a special routine which identifies identical gates and renames them as necessary. The routine also identifies imbedded identical gates.

LOGICAL ANALYSIS

The purpose of logical analysis is a qualitative assessment of whether, or to what degree, common events impaired the design redundancy of the system. The analysis is done by reduction of common events through restructuring of the fault tree. Some structural changes modify the meaning of certain composite events defined by logical gates. To facilitate interpretation of those changes, reduction can proceed in steps. Details are given in ref. 9.
FIGURE 31
Erasure of inversed events
FIGURE 32
Identification of Identical Gates
An important part of logical analysis is the evaluation of effects a failure has on the decrease of redundancy in a system. Such an evaluation is used e.g. for decisions whether a repair of a failed equipment can be postponed without jeopardizing safety, or whether an immediate repair is necessary. An event which already has happened is identified in the fault tree by a HOUSE - Fig. 33. It can be defined either by an appropriate entry in the basic description block of the fault tree description file, or interactively using the command HO. The interactive definition is used mainly for quick familiarisation with the fault tree behaviour.

The logical analysis routine is called on the HP-1000 computer by commands TR, FTL. It can be called from any terminal including remote access through a 300 baud modem.

If enough computer memory is available, the logical reduction routines can be called also from the display plotting program FTP.

**NUMERICAL EVALUATION**

The fault tree is a suitable tool for evaluation of probabilities of random independent events (failures). The rules for calculation of probabilities are in this case quite simple. Probability of an event defined by an AND-gate is obtained by multiplying the probabilities of input events. Probability of an event defined by an OR-gate is obtained by adding the probabilities of input events and subtracting a factor which corresponds to the probability of input
Event which has already happened (HOUSE)
events occurring together. If the probabilities are small, this factor may be neglected.

The apparent ease with which the evaluation can be done encourages the use of fault trees even in other cases. The result can be mathematically correct or the errors may be so small that practical usefulness of results is not affected. Unfortunately, in some cases the results of a seemingly correct calculation can be wrong by several orders of magnitude. Use of a computer may aggravate the problem considerably because in a complex calculation of a large tree it is almost impossible to identify an error once it has occurred. Special precautions are therefore required.

The AECB program prevents large errors by: explicit definition of category of events on the basic event description card, verification of logical structure, and the rejection of constructs which cannot be dependably evaluated by the fault tree method.

The easy with which even large trees can be handled eliminates the need for premature truncation and this considerably decreases the number of missed common events which are a major source of errors. In order to demonstrate the effect of this improvement, Fig. 34 show an example of a fault tree with calculation of probabilities based on the assumption of independent random events. The diamonds on events K and L indicate that the tree was truncated. The truncation obfuscated that event B is common to P, K and L. The correct probabilities obtained by analysis of the complete tree are on Fig. 35.
FIGURE 34
Calculation of probabilities when common event missed

FIGURE 35
Correct calculation of Probabilities
Prevention of other sources of errors, which include mixing of units (failure rates, unavailabilities) and large deviations from randomness (e.g. effects of maintenance policies, queuing etc.) necessitated some restrictions of acceptable logical structures. Input events for AND-gates must be defined by unavailability or failure probability. Use of failure rates is in this case not acceptable. On the other hand, the use of WHILST is restricted so that the output event of a WHILST gate must be a failure identified by failure rate and not unavailability or failure probability. NOT can be used only for events indicating states, that means events identified by availabilities or failure probabilities.

The main purpose of the numerical evaluation section of the computerized fault tree analysis system is a verification that common events do not degrade unacceptably the reliability and safety of the analysed system. The results can be also used for identification of those systems or their parts which require further analysis using other more rigorous methods. However, the whole system is intended as a tool and not a "question-answering-machine". The usefulness of the system depends on its user.

The numerical evaluation routine is called by the command TR,FTE. The evaluation starts with a direct calculation of the original tree followed by direct calculation after reduction of common events. If the structure of the tree with common events does not allow a full reduction, the residual tree is processed by simulation. The output can be directed either to the terminal (command TR,FTE) or to the printer (command TR,FTE,6). The direct calculation
of probabilities gives maximum attainable precision, however in the process of reduction of common events some composite events (except the top event) may be redefined or skipped. The program therefore offers the possibility to use directly the simulation for evaluation of the original tree. The command is RU, FTE,6,S or RU,FTE,,S. Simulation offers only limited precision, but evaluates all events as defined in the fault tree description file.

The internal coding scheme of probability values leads, for a given simulation, to a trade-off between precision and range of values. With respect to the prevailing low accuracy of input data it was considered acceptable to use a discrimination factor of two, which means that failure rates or unavailabilities must differ by a factor of more than 2 to be coded differently. This choice leads to a range of probabilities (or unavailabilities) of basic events from $10^{-1}$ to $10^{-5}$ and subsequently to failure rates between 10 failures/year to 1 failure in 1000 years ($10^{-3}$ failures/year).

The simulation proceeds in sequences representing 160 years each. Results of each sequence are displayed on the terminal. The simulation terminates on the count of occurrences of the top event. If required, simulation can be terminated prematurely by the RTE-IV system command BR,FTFTE. In that case the program completes the sequence and terminates execution by printing the results.

**PRINCIPLE OF OPERATION**

Users of the fault tree analysis system do not need to be concerned with
the internal operation of the system. A short description of the principles of operation is included in this report for reliability engineers and computer specialists working in the field of fault tree analysis.

Most operations, including error checking and ordering of input cards, are quite ordinary and do not need explanation. The distinguishing feature of this system is handling of the tree in the form of a string, which is the key to the pattern recognition abilities of the system.

The event string is described by five arrays: NCODE, TYPE, TROUT, EVUP and EVDN.

The array NCODE contains reference numbers for codes and names of events stored in the auxiliary file on disk. The reference numbers are assigned in that sequence in which event codes are encountered in the fault tree description file. Reference number 1 is assigned to the title. The types of events are stored in the array TYPE. Common events occupy as many array locations as there are occurrences of the event in the tree. However, all locations are identified with the same reference number in NCODE.

The array TROUT identifies category of events. It is assigned by bits. Bit 12 (decimal 4096) is set if event identifies a condition (numerical value is defined by probability or unavailability). Bits 13 and 14 are both cleared if event is independent (occurs in the tree only once). For common events bits 13 and 14 identify the position of the particular events in the string of events
with the same reference number.

OI (8192-16383) leading common event
IO (16384-24576) common event
II (24576-32767) transfer-out

Other bits are unassigned and can be used locally. However, if a subroutine uses bits 14 or 15 (32768 or sign), it must clear them before returning to the calling program.

EVUP and EVDN contain references to the location of the event above (gate event) and below (inputs to the gate). For basic events EVDN is zero.

The top event is stored as the first event of the string. Other events follow in the same order as listed in the plot control file. The string does not contain an explicit indication of the level in which an event is located. This information is required for plotting and is generated locally when needed. To simplify some operations, location of end events in each level is stored in the array LEVRM. Information in LEVRM stays valid only as long as the event string did not change.

For plotting and printing the tree additional information on event coordinates is required. Two algorithms were developed for assigning the coordinates. Algorithm SSB assigns a separate slot (24 units) for each basic event. Assignment starts with finding the leftmost basic event and continues to the right. Coordinates of the gates are obtained afterwards by interpretation. "Separate slots" algorithm is short and fast. The separate slots are usefull
in some special cases when it is desirable to outline fault tree partitions (e.g. in theoretical work on pattern recognition). However, the separate slots make the presented tree less compact. The basic algorithm of the system is therefore the SBL algorithm which reserves the slot for a basic event only on the level of the basic event and one level below. The assignment starts with assignment of slots in the last level, which must contain only basic events. In the levels above the assignment depends on whether the event is a gate or a basic event. Coordinates of basic events are assigned by interpolation. For basic events the required slot is squeezed between events on one level down. This may require a shift of already assigned coordinates to the right, as the width of a basic event slot must be 24 units.

Numerical calculation processes probabilities directly. Failure rates are internally stored and processed as probabilities of failures within a trial period of 1/100 year. This scale (in subroutines PTRANS, PTRAN) limits the maximum failure rate to about 10 failures per year. Formulas for numerical evaluation are in subroutine PTEG.

Simulation controlled by subroutine PTRANS works with bits. A set bit corresponds to a failure. Unavailability is simulated by failures in a proportionate number of trial periods (0.01 year). On HP-1000 computer with 16 bit words, a pass corresponds to 16 trials. One trial sequence consist of 1000 passes representing 160 years. The probabilities of failures of basic events can be between $3/32$ (approx. 0.1) and $3/2^{18}$ (approx. $10^{-5}$), corresponding with the selected trial period of 1/100 to failure rates between 10 failures.
per year to 1 failure in 1000 years. The discrimination factor is 2, which means that failure rates or unavailabilities must differ by a factor of more than 2 to be coded differently.

SOFTWARE PACKAGE

The operation of the fault tree analysis system is controlled by four execution control files (transfer files), so that the user does not need to be aware of the complexity of the software package with six main programs and 52 subroutines.

All programs and subroutines are written in FORTRAN IV for HP-1000. The listing contains 5400 lines. Source code is stored on five HP9162-0061 data cartridges (cassettes) on 700 feet of tape (at 800 BPI). To facilitate handling, the 64 subroutines were grouped in seven library files. Five loader control files facilitate loading.

The maximum size of the fault tree to be plotted or analysed depends on the dimensioning of data arrays in the main programs, which can be defined to fit the available computer memory. Subroutines use variable dimensions for data arrays and do not require modification.

The six main programs are FTFTI, FTFTL, FTFTP, FTFTB, FTFTE and FTFTZ.

Program FTFTI reads the data in the fault tree description file and checks the input for format errors and logical inconsistencies. Afterwards it orders
fault tree events in a string and creates an auxiliary unprintable file in which it stores the fault tree description in a form suitable for further processing.

The program call includes three arguments (e.g. RU,FTFTI,11,2,4). The first argument defines where eventual error in input files should be listed. When omitted (as in the example), the errors are listed on the terminal from which the program was entered. The second argument defines the operating mode. Mode 1 is used for numerical evaluation. The numerical data for basic events is read. The logical structure of the fault tree is checked for acceptability by the numerical evaluation routine. Mode 2 is used for plotting and logical analysis. In this mode no numerical data is read and the check of logic is limited to a check of inconsistent use of WHILST and common events. The types of gates (e.g. A, R, X) and events (e.g. D, E, H), which can be used in mode 1 and 2 is defined in the program by the content of variable arrays GATCH and BEVCH which are fixed by DIMENSION and DATA statements in the main program.

The third argument defines the size of the tree for storing in the auxiliary file. The number defines the number of blocks with 120 events each. The maximum is given by dimensioning of arrays NCODE, TYPE, EVUP, EVDN, TROUT which have a size MXEVD corresponding to the maximum number of events, and arrays EVBIN, GATE, NINF, BIN(6,n), CHECK which have a size MXCRD corresponding to the maximum number of gate description lines (cards) in the fault tree description file. For operation in a 18 k-word memory partition the third argument can not exceed 4 because the arrays are set for 480 events described on 240 gate description cards. A memory partition of 32 k-words could
accommodate 2160 events (argument 1 & 2). The maximum number of levels of the tree is determined by the dimension MXLEV of array LEVRM, which is set to 40. Maximum is 117, limited by the structure of the auxiliary file.

The auxiliary file is type 1 (direct access) file with the name FT"n where n depends on the terminal from which the program was called. The auxiliary file has several sections. The first section contains codes of events. The second section contains names of events. To save memory space neither codes nor names are stored in arrays and they are always read from the auxiliary file wherever needed. The third section of the auxiliary file contains logical description of the tree, which is, during processing, stored in memory. This section of the auxiliary file serves only for purposes of communication between programs. The fourth section stores numerical data and is used only in mode 1 (numerical processing). If the FTFTI program could not create a usable auxiliary file, for any reason it returns global parameter 2P with a value 99.

Program FTFTP is fault tree graphics routine. The type of input used for processing depends on the arguments in the RU command. If the argument is omitted (command RU, FTFTP), the program asks for the name of the plot control file which should be plotted. If the value of the second argument is non zero but less than 9, the program reads input data from the auxiliary file with the name FT"n. If the input is in the format of an auxiliary file, but with another name, the name of the file is used in forming the arguments (e.g. RU, FTFTP,DE,MO,F4 for input file DEMOF4).
The program has two modes of operation selected by interactive commands: display and plotting. In the display mode the whole fault tree (or its segment) can be inspected by scanning. Certain graphical editing is also possible in this mode. When plotting is required on a remote plotter (for large trees), a plot control file can be created. For plotting on the desk top plotter the plotting mode of the program allows the user to position the plotting window to define which portion of the fault tree should be plotted and how it should be positioned. A command initiating plotting is also issued in this mode. After program termination the last displayed portion of the fault tree is retained in the graphics memory of the 2648A terminal and can be restored on the screen. This proved to be advantageous when changes in the fault tree description file are required, as both the fault tree and the edited description file can be displayed simultaneously.

When the available memory partition is only 18 k-words, the size of the fault tree is limited to 120 events and the program is not capable of positioning the title on the plot, and a separate plotter control routine L77 has to be used. The limited available memory also does not allow simultaneous use of logic reduction routines. The logic reduction routines require 4 k-words, the title routine 1 k-words, and large trees an additional 1 k-words for each block of 120 events.

Program FTFTB is designed for sectioning a large fault tree in segments. The large fault tree had to be processed by FTFTI program and its description stored in the format of an auxiliary file. The name of this file is arbitrary and it is transferred in the form of arguments in the RU command (e.g. RU,
PTTB, FT,"", #1). The description of the selected branch of the tree is stored in a new auxiliary file with the standard name FT"&n. If for any reason the new auxiliary program was not created, the program returns global parameter 2P with the value 99, same as PTFTI program. When operated in a 18 k-word partition, the program can section trees with 480 events in branches with up to 120 events.

Program PTFTL is used for logical analysis of fault trees. It can be called by non-graphics terminals. No parameters are passed. The input is from an auxiliary file with the standard name FT"&n. When operated in a 17 k-word partition, the program can handle trees with up to 480 events. Additional 1 k-word of memory is required for each new block of 120 events.

Program PTFTE is used for numerical evaluation of probabilities, unavailables and failure rates. The input is from an auxiliary file with the standard name FT"&n. The operation is controlled by parameters in the RU command. The first parameter defines whether the output should be printed (parameter 6) or just displayed (parameter omitted). The second parameter defines whether the evaluation should include an attempt for direct calculation (parameter omitted) or just random simulation (parameter S, e.g. RU,PTFTE,6,S). Simulation can be prematurely terminated by the RTE-IV system command BR, PTFTE. The size of the fault tree which can be evaluated is limited by the dimensioning of data arrays which have to fit the available memory space. For 18 k-word memory partition the limit is 240 events. Each additional groups of 120 events requires 1.2 k-words of memory.
Program **FTFTZ** is a support routine used for purging the auxiliary file at the end of processing.

**Library files** are named according to the first subroutine in each file.

%PTINL - input routines for fault tree logic;
%FTIND - input routines for numerical data;
%FTSBL - general fault tree handling routines;
%FTRET - fault tree reduction routines;
%FTJGCC - plotting routines;
%FTSWE - on-screen editing of fault tree graphics;
%FTPND - numerical evaluation routines.

When loaded, the library file %FTIND must be called before %PTINL and the library file %FTRET must be called before %FTSBL to prevent marking by dummy routines FTIND and FTRET which were incorporated in order to enable creation of smaller systems with only basic capabilities for computers with very limited memory space.

**Conversion to other computers** than HP-1000 will require some modifications of the source code due to differences in "dialects" of FORTRAN IV. The main changes will be in disk file operation where HP-1000 does not accept READ and WRITE statements and requires use of special system routines with direct control of buffers. The display and plotting routines - with exception of the plot control file creation routine - are device dependent and would have to be rewritten if other graphics terminals and desk top plotters are used.
At the time of writing this report a fault tree plotting routine for processing plot control files was already operational on CDC 6000 controlling CALCOMP plotters at CRNL's Computing Centre at Chalk River, Ontario and at the Department of Energy, Mines and Resources in Ottawa. The name of that program is FTPLT.
ECB-SED RELIABILITY INFORMATION SYSTEM

NUMERICAL ANALYSIS OF FAULT TREE:

HYPOTHETICAL EXAMPLE

8 JAN., 1978

REVIEW OF INPUT DATA

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAF</td>
<td>LIGHT ALARM FAILED</td>
</tr>
<tr>
<td>DCP</td>
<td>DC POWER UNAVAILABLE</td>
</tr>
<tr>
<td>REC</td>
<td>POWER FROM RECTIFIERS UNAVAILABLE</td>
</tr>
<tr>
<td>GPU</td>
<td>POWER FROM GENERATORS UNAVAILABLE</td>
</tr>
<tr>
<td>MPU</td>
<td>LOSS OF MAIN POWER</td>
</tr>
<tr>
<td>BPU</td>
<td>LOSS OF BACK-UP POWER</td>
</tr>
<tr>
<td>GSC</td>
<td>GENERATOR CONTROL UNAVAILABLE</td>
</tr>
<tr>
<td>GEN</td>
<td>GENERATORS FAILED TO OPERATE</td>
</tr>
<tr>
<td>HBU</td>
<td>BACK-UP 125V UNAVAILABLE</td>
</tr>
<tr>
<td>BPS</td>
<td>BACK-UP POWER SUPPLY UNAVAILABLE</td>
</tr>
<tr>
<td>ACF</td>
<td>AUTOMATIC CONTROL FAILURE</td>
</tr>
<tr>
<td>HMU</td>
<td>125V MAIN UNAVAILABLE</td>
</tr>
<tr>
<td>LAR</td>
<td>LIGHT ALARM REQUIRED</td>
</tr>
<tr>
<td>MPS</td>
<td>MAIN POWER SUPPLY FAILURE</td>
</tr>
<tr>
<td>MCF</td>
<td>MANUAL CONTROL FAILURE</td>
</tr>
<tr>
<td>GS1</td>
<td>GENERATOR 1 UNAVAILABLE</td>
</tr>
<tr>
<td>GS2</td>
<td>GENERATOR 2 UNAVAILABLE</td>
</tr>
<tr>
<td>GS3</td>
<td>GENERATOR 3 UNAVAILABLE</td>
</tr>
<tr>
<td>PSB</td>
<td>POWER SUPPLY FAILURE</td>
</tr>
<tr>
<td>PSD</td>
<td>POWER SUPPLY NOT SWITCHED ON</td>
</tr>
<tr>
<td>CEF</td>
<td>CONTROL EQUIPMENT FAILURE</td>
</tr>
<tr>
<td>HPU</td>
<td>OUTSIDE POWER UNAVAILABLE</td>
</tr>
<tr>
<td>SWY</td>
<td>SWITCHYARD FAILURE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>.25F./YEAR (2.9E-05F./HR)</td>
</tr>
<tr>
<td></td>
<td>9.0E-05</td>
</tr>
<tr>
<td></td>
<td>2.0E-02</td>
</tr>
<tr>
<td></td>
<td>5.0E-02</td>
</tr>
<tr>
<td></td>
<td>5.0E-02</td>
</tr>
<tr>
<td></td>
<td>1.5E-03</td>
</tr>
<tr>
<td></td>
<td>1.0E-02</td>
</tr>
<tr>
<td></td>
<td>9.1E-05</td>
</tr>
<tr>
<td></td>
<td>2.3E-04</td>
</tr>
<tr>
<td></td>
<td>1.4E-04</td>
</tr>
</tbody>
</table>

FIGURE 36A

Numerical evaluation by calculation
AECB-SED RELIABILITY INFORMATION SYSTEM

DIRECT NUMERICAL EVALUATION OF INDEPENDENT GATES

BPS  BACK-UP POWER SUPPLY UNAVAILABLE  1.1E-02
GEN  GENERATORS FAILED TO OPERATE  7.2E-03

FIGURE 36B

Numerical evaluation by calculation
AECB-SED RELIABILITY INFORMATION SYSTEM

REDUCED FAULT TREE

---
LAF
W
---
LAR
E
DCP
R
---
REC
R
---
GPU
R
---
HMU
25
GSC
A
GEN
A
---
HPU
SWY
MPS
BPS
CEF
MCF
C
L
R

NUMERICAL EVALUATION OF REDUCED TREE

<table>
<thead>
<tr>
<th>Event</th>
<th>Failure Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>GSC GENERATOR CONTROL UNAVAILABLE</td>
<td>1.8E-36</td>
</tr>
<tr>
<td># # F # 25</td>
<td>1.0E-06</td>
</tr>
<tr>
<td>HMU 125V MAIN UNAVAILABLE</td>
<td>3.7E-04</td>
</tr>
<tr>
<td>GPU POWER FROM GENERATORS UNAVAILABLE</td>
<td>7.3E-03</td>
</tr>
<tr>
<td>REC POWER FROM RECTIFIERS UNAVAILABLE</td>
<td>3.7E-04</td>
</tr>
<tr>
<td>DCP DC POWER UNAVAILABLE</td>
<td>7.6E-03</td>
</tr>
<tr>
<td>LAF LIGHT ALARM FAILED</td>
<td>1.9E-03F./YEAR</td>
</tr>
</tbody>
</table>

FIGURE 36C

Numerical evaluation by calculation
<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAF</td>
<td>Light Alarm Failed</td>
<td>2.7E-03F./Year</td>
</tr>
<tr>
<td>LAR</td>
<td>Light Alarm Required</td>
<td>2.9F./Year (3.3E-05F./hr)</td>
</tr>
<tr>
<td>DCP</td>
<td>DC Power Unavailable</td>
<td>6.4E-03</td>
</tr>
<tr>
<td>REC</td>
<td>Power from Rectifiers Unavailable</td>
<td>3.4E-04</td>
</tr>
<tr>
<td>GPU</td>
<td>Power from Generators Unavailable</td>
<td>6.1E-03</td>
</tr>
<tr>
<td>MPU</td>
<td>Loss of Main Power</td>
<td>4.0E-04</td>
</tr>
<tr>
<td>BPU</td>
<td>Loss of Back-Up Power</td>
<td>1.3E-02</td>
</tr>
<tr>
<td>OSC</td>
<td>Generator Control Unavailable</td>
<td>1.9E-05</td>
</tr>
<tr>
<td>GEN</td>
<td>Generators Failed to Operate</td>
<td>6.1E-03</td>
</tr>
<tr>
<td>MPS</td>
<td>Main Power Supply Failure</td>
<td>9.2E-05</td>
</tr>
<tr>
<td>H3U</td>
<td>Back-Up 125V Unavailable</td>
<td>3.4E-04</td>
</tr>
<tr>
<td>BPS</td>
<td>Back-Up Power Supply Unavailable</td>
<td>1.3E-02</td>
</tr>
<tr>
<td>ACF</td>
<td>Automatic Control Failure</td>
<td>4.4E-04</td>
</tr>
<tr>
<td>MCF</td>
<td>Manual Control Failure</td>
<td>2.3E-02</td>
</tr>
<tr>
<td>GS1</td>
<td>Generator 1 Unavailable</td>
<td>4.7E-02</td>
</tr>
<tr>
<td>GS2</td>
<td>Generator 2 Unavailable</td>
<td>4.7E-02</td>
</tr>
<tr>
<td>GS3</td>
<td>Generator 3 Unavailable</td>
<td>4.7E-02</td>
</tr>
<tr>
<td>PSB</td>
<td>Power Supply Failure</td>
<td>1.5E-03</td>
</tr>
<tr>
<td>PSD</td>
<td>Power Supply NOT Switched On</td>
<td>1.2E-02</td>
</tr>
<tr>
<td>HMB</td>
<td>125V Main Unavailable</td>
<td>3.4E-04</td>
</tr>
<tr>
<td>CEF</td>
<td>Control Equipment Failure</td>
<td>9.2E-05</td>
</tr>
<tr>
<td>HPU</td>
<td>Outside Power Unavailable</td>
<td>1.8E-04</td>
</tr>
<tr>
<td>SHY</td>
<td>Switchyard Failure</td>
<td>1.8E-04</td>
</tr>
</tbody>
</table>

**FIGURE 37**

Numerical evaluation by simulation
REFERENCES


APPENDIX 1

LIST OF COMPUTER Routines

1. Main Programs
   FTPTI - Read fault tree description file
   FTPTL - Interactive processing for non-graphics terminal
   FTPTP - Display fault tree and plot it
   FTPTB - Processing large trees in branches
   FTPTE - Numerical evaluation of fault trees
   FTPTZ - Purge auxiliary file

2. Library Files

&FTINL - Input Subroutines
   FTPNL - Read fault tree description - gates
   FTECK - Check if all cards used
   FTPND - Read cards in data block - short version
   FTPCD - Read data line in disc file and convert to 80 char
   FTPDC - Convert gate description card in standard format
   FTPKN - Find token after L, shift it to M, and update L
   FTPDES - Encode event starting at L and return sequence number
   FTPNMD - Interpret standard name card and store name in file
   FTPTES - Order tree events in a tree
   FTPRCN - Reverse categories of numerical data

&FTIND - Subroutines for Numerical Data Input
   FTPND - Read cards in data block
   FTPRD - Read reliability data from input card
   FTPNRN - Read floating point number from array of 80 char.
   FTPIFA - Read integer number from array of 80 char.

&FTSBL - Basic Fault Tree Processing Routines
   FTPSBL - Assign coordinates
   FTPCF - Create plot control file
   FTPRS - Print fault tree skeleton
   FTPTA - Turn tree coordinates so that left is top
   FTPCOD - Retrieve code from auxiliary file
   FTPNAM - Retrieve name from auxiliary file
&FTSBL - Basic Fault Tree Processing Routines (cont'd)

FTEVP - Find code in auxiliary file and return its sequence number
FTDAF - Store arrays with fault tree description in auxiliary file
FTGAF - Get arrays with fault tree description from auxiliary file
FTRET - Dummy reduction routine (to save memory)
FTBGS - Display string of events for debugging

&FTRET - Fault Tree Reduction Routines

FTRET - Common event reduction - calling subroutines
FTNTR - Reduce structures with NOT gates
FTTSG - Advance N to top of string of gates ITYP
FTHSR - Remove HOUSE events and impossible events
FTMSH - Mark sister HOUSES and impossible events
FTFIG - Find gates with identical common events
FTSAC - Look for erasing and stepping stones in alternate chains
FTRER - Reduce fault tree by removing DR-of-OR's and AND-of-AND's
FTREA - Reduce fault tree by removing AND-of-OR's and OR-of-AND's.
FTUAM - Unification algorithm for masking
FTRGS - Remove double input gate which will become single input.
FTGBE - Look for gates with independent basic events only.
FTSES - Sort events in a string.
FTMEV - Move event up from L to N.
FTRCG - Remove M events starting at N and close the gap.
FTDIG - Insert dummy gate
FTDIG - Remove inputs to gate IGAT
FTTRS - Transfer references of transfer-out to upper sister.
FTSTR - Mark names of modified events with a star.

&FTCGT - Graphics Subroutines

FTCGT - Control of fault tree display on terminal
FTCDP - Control of desk top plotter from terminal
FTGCC - Graphics control commands
FTPGT - Plot fault tree gate on graphics terminal 2648A
FTGCS - Plot on 2648A Terminal pattern from BIN
FTLIS - Convert integer to ASCII string.
FTPGP - Plot fault tree gate on AECB plotter HP-9872A
FTPGC - Incremental coordinates of gates
FTPRS - Dummy fault tree printing routine (to save memory)
&PTSWE - Subroutines for Graphical Edit

FTSWE - Switch events in a displayed gate
FTMSG - Mark switched gates with numbers
FTHSI - Interactive input of house events
FTIEL - Insert extension line
FTCGT - Get code from terminal and locate in event string.

&FTPND - Numerical Evaluation Routines

FTPND - Print line with numerical data for event ICOD.
FTEGI - Numerical evaluation of gates with independent events.
FTSRG - Simplified simulation of random failures
FTIRN - Integer random number routine
MWCBW - Count bits in a word
FTGPM - Evaluate failure pattern for a gate.
IBINC - Calculate binomial coefficients.
SCOMP - Sum of combinational probabilities.

3. Loader Control Routines

FTFXI - Loading of FTFTI
FTFXL - Loading of FTFTL
FTFXP - Loading of FTFTP
FTFXB - Loading of FTFXB
FTFXE - Loading of FTFXE

4. Routines for External Plotting

FTPLT - Plot fault tree from control file (main)
FTITP - Plotter initialization routine
FTTIL - Determine length of title
FTTIT - Plot title
FTPST - Plot fault tree event
FTLIN - Plot a line from array of coordinates
APPENDIX 2

LIST OF INTERACTIVE COMMANDS FOR HP-1000

Syntax of commands corresponds to that used by RTE-IV operating system. Parameter indicated by n follows a single letter command immediately (e.g. L4), but it is separated from a double letter command by a comma. Optional parameters are shown in brackets.

1. General Commands

For use by both graphics and non-graphics terminals. Selection of commands actually implemented depends on available memory, e.g. at AECE with 18 k-words memory, the command RE can be called only from FTL but not from FTP routine; and commands SS and TR are not implemented at all.

EX - exit from program
FI - create plot control file
HO - define event which already happened (house)
LI - insert an extension line
NW - new load of original data from auxiliary file
PR(n) - print fault tree skeleton on logical unit n
RE(n) - reduce common events
SB - use basic algorithm for assigning coordinates
SI - display size of present tree
SQ(n) - squeeze, truncate fault tree at nth level
SS - assign each basic event to a separate slot
ST - store present fault tree in a binary file
TR - truncate inputs to a gate
; - cancel command requiring answer.

2. Plotting Commands

The display and plotting routine FTP uses four modes of operation, with different commands. EXit can be used in all modes. Meaning of "move" commands depends on mode.

a. display mode

mm - change plotting format
   m1 - codes and names
   m2 - codes only
   m3 - truncate names only
PL(n) - same as WI(n)
WI(n) - pick up pen n and switch to plot window mode
X(n) - shrink display by n steps
Z(n) - zoom, magnify display by n steps
;(n) - display frame for magnification by n steps
and switch to zoom frame mode
general commands can be used also. Move commands (2.d) refer to display.

b. zoom frame mode:

   z - zoom, magnify as shown by frame
   ; - return to display mode
   move commands refer to zoom frame

c. plot window mode

   PL - plot as shown by window
   WI - define new window for a different size of paper
   X - use small plot format (for M1 only)
   ; - return to display mode
   move commands refer to plot window. Parameters
   may contain decimal fractions.

d. move commands

   L(n) - move left by n levels
   R(n) - move right by n levels
   /(n) - move down by n events
   \(n) - move up by n events