PAFT F77
PROGRAM FOR THE ANALYSIS OF FAULT TREES
[September 1982, Fortran 77 Version]

Operations Research Center Research Report No. 83-14

Thomas A. Feo

October 1983

U. S. Army Research Office - Research Triangle Park

DAAG29-81-K-0160

Operations Research Center
University of California, Berkeley

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

83 11 15 117
THE VIEW, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHOR(S) AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY, OR DECISION, UNLESS SO DESIGNATED BY OTHER DOCUMENTATION.
REPORT DOCUMENTATION PAGE

<table>
<thead>
<tr>
<th>REPORT NUMBER</th>
<th>1. TECR NUMBER</th>
<th>SECURITY CLASSIFICATION OF THIS PAGE (WHEN DATA ENTERED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORC 83-14</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE (and subtitle)</th>
<th>5. TYPE OF REPORT &amp; PERIOD COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM FOR THE ANALYSIS OF FAULT TREES</td>
<td>Research Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7. AUTHOR(s)</th>
<th>6. PERFORMING ORG. REPORT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thomas A. Feo</td>
<td>DAAG29-81-K-0160</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>9. PERFORMING ORGANIZATION NAME AND ADDRESS</th>
<th>10. PROGRAM ELEMENT, PROJECT, TASK AREA &amp; WORK UNIT NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operations Research Center</td>
<td>P-18195-M</td>
</tr>
<tr>
<td>University of California</td>
<td></td>
</tr>
<tr>
<td>Berkeley, California 94720</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>11. CONTROLLING OFFICE NAME AND ADDRESS</th>
<th>12. REPORT DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>U. S. Army Research Office</td>
<td>October 1983</td>
</tr>
<tr>
<td>P.O. Box 12211</td>
<td></td>
</tr>
<tr>
<td>Research Triangle Park, North Carolina</td>
<td></td>
</tr>
<tr>
<td>27709</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13. NUMBER OF PAGES</th>
<th>14. MONITORING AGENCY NAME &amp; ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15. SECURITY CLASS. (OF THIS REPORT)</th>
<th>15a. DECCLASSIFICATION/DOWNGRADING SCHEDULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
<td></td>
</tr>
</tbody>
</table>

16. DISTRIBUTION STATEMENT (OF THIS REPORT)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (OF THE ABSTRACT ENTERED IN BLOCK 20, IF DIFFERENT FROM REPORT)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (CONCEDE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

Fault Tree
Marginal Importance
Occurrence Rate
Marginal Occurrence Rate

Replicated Event
Basic Event

20. ABSTRACT (CONCEDE ON REVERSE SIDE IF NECESSARY AND IDENTIFY BY BLOCK NUMBER)

(SEE ABSTRACT)
ACKNOWLEDGEMENT

I would like to express my gratitude to Professor Richard E. Barlow for his help, encouragement, and insight throughout the formulation of this work.

Further acknowledgement is due to Thomas A. Barlow and Kevin Wood for their original work on PAPT during the summer of 1981.
PAFT F77, a program for the analysis of fault trees coded in Fortran 77, is presented. Given the structure of a fault tree and the probability or failure rates of its basic events, PAFT F77 calculates the probabilities of the top and all intermediate events, as well as the marginal importances of all basic events. When the input is in terms of failure rates instead of probabilities, the marginal occurrence rates of all basic events and the occurrence rate of the top event are also calculated. This additional information though, is invalid for fault trees comprising NOT gates or those gate types that must be modeled with NOT gates such as exclusive OR gates. The program is designed to handle simple fault trees and those with replicated basic events.

The introduction to this text describes the program's ability and limitations together with a brief theoretical review of fault tree analysis. The second section is a users guide to PAFT F77 on the UNIX operating system at the University of California, Berkeley. The final section gives an in depth description of the program's Fortran 77 code.
TABLE OF CONTENTS

1. INTRODUCTION TO PAFT F77
   1.1 What is a Fault Tree? ........................................ 1
   1.2 The Program's Ability ....................................... 2
   1.3 Theoretic Background ....................................... 3
   1.4 The Program's Limitations ................................. 5

2. USER'S GUIDE
   2.1 Using PAFT F77 on UNIX .................................... 6
   2.2 Input File Structure ...................................... 6
   2.3 Output File Examples .................................... 14

3. PAFT F77 LISTING DESCRIPTION
   3.1 Overview of Program Structure ........................... 18
   3.2 Global Variable Definitions ............................. 20

APPENDIX - PAFT F77 Code ..................................... 22
REFERENCES .................................................. 35
1.1 What is a Fault Tree?

The safety of nuclear reactors is one of the many engineering risk considerations in the forefront of today's political, environmental and social dialogue. Answering these questions of safety from an emotional or political point of view falls short of the scientific objectivity which our society expects and demands. Fault tree analysis is a method available to the engineer for determining reliability of complex systems. A fault tree is a directed acyclic graph representing the relationship between a system event and subsystem events. In turn, each subsystem event can be represented by more detailed subsystem events until fundamental or basic events are reached. The state of each fundamental component or basic event will determine the success or failure of the entire system, through the logic structure of the fault tree.

Constructing a fault tree for a complex system can be a formidable task. The millions of components of the system must be identified and mathematically modeled. Then this data must be used to construct a fault tree which correctly represents the logical dependence between the system's component events. For the fault tree of a nuclear reactor, the top event could be the catastrophic occurrence of core meltdown, while a basic event could be as insignificant as the outside humidity. Fault tree analysis is not limited to nuclear reactor reliability but is a commonly used statistical and engineering tool. The following example should give some idea of the variety of applications of this method.

You are offered a game of chance involving a fair coin. If either of the first two flips is a head and either of the second two flips is a head
you lose $1, otherwise you win $1. In this example, the top event is not a
catastrophe, but still the basic principles of probability behind this game
of chance can be applied to more serious and consequential cases. The fault
tree (in this case it would more likely be called a game tree) is:

```
          Lose $1
             | AND gate
             |     OR gate
             |     OR gate
Heads    Heads    Heads    Heads
Basic event #1  Basic event #2  Basic event #3  Basic event #4
```

The probability of each occurrence of a head is \( \frac{1}{2} \). The probability of
each OR gate is \( \frac{3}{4} \) and the probability of the AND gate or top event is \( \frac{9}{16} \).
Therefore, with the opportunity quoted, one would gracefully decline this
game since your expected winnings are \( -\frac{1}{8} \) of a dollar.

1.2 The Program's Ability

The input to PAFT F77 is the structure of a fault tree together with
the probabilities or failure rates of its basic events. The program returns
the probabilities of all gate events including the top event and the marginal
importances of all basic events. The marginal importance of a basic event
is the partial derivative of the probability of the top event with respect
to the probability of the basic event. The marginal importance can also be
interpreted as the probability of the top event given that the basic event
has occurred, minus the probability of the top event given the basic event
has not occurred. When analysing a system, it is useful to know which basic
events most influence the occurrence of the top event. The marginal importance is one measure of such influence.

When the input is in terms of failure rates instead of probabilities, the marginal occurrence rates of all basic events and the occurrence rate of the top event are also calculated. The marginal occurrence rate of a basic event, like its marginal importance, is a measure of the influence a basic event has on the top event. The occurrence rate of the top event is the sum of all of the marginal occurrence rates of the basic events. When a fault tree contains NOT gates or those gate types that must be modeled with NOT gates such as exclusive OR gates, the program, which assumes monotonicity of the top event with respect to basic events, calculates the marginal occurrence rates of the basic events under these gates incorrectly. Furthermore, the sum of these quantities produces an incorrect occurrence rate for the top event. Hence, marginal occurrence rates may be of little value for fault trees with NOT gates and exclusive OR gates.

1.3 Theoretical Background

The following functions are used to calculate the probabilities of AND, OR, EXOR, and NOT gates.

\[
P_{\text{AND}} = \Pi p_i \\
P_{\text{OR}} = 1 - \Pi (1 - p_i) \\
P_{\text{EXOR}} = p_1 + p_2 - 2p_1p_2 \\
P_{\text{NOT}} = 1 - p .
\]

The remaining gate types (such as two out of three \((2/3)\) or exactly 5 out of 7 \((x5/7)\)) use the following generating function.
\[ \prod_{i=1}^{n} (q_i + p_i z) . \]

For the two out of three case, the sum of the coefficients of \( z^2 \) and \( z^3 \) is the correct probability with \( n = 3 \); for the event exactly 5 out of 7, the coefficient of \( z^5 \) is the correct probability with \( n = 7 \).

The marginal importance of a basic event is

\[ P(\text{top} \mid \text{basic event}) - P(\text{top} \mid \text{not basic event}) . \]

The marginal occurrence rate is (at a specified time \( t \))

\[ \text{(failure rate of basic event)} \times \text{(marginal importance of basic event)} \times \frac{(1 - P(\text{basic event}))(1 - P(\text{top}))}{1 - P(\text{top})} . \]

If \( r_i(t) \) is the occurrence rate of basic event \( i \) at time \( t \), \( F_i(t) \) is the probability event \( i \) occurs (for the first time) by time \( t \), \( I_i(t) \) is the marginal importance of basic event \( i \) at time \( t \) and \( P(T > t) \) is the probability the top event has not occurred by time \( t \), then the basic event occurrence rate at time \( t \) is

\[ r_i(t)[1 - F_i(t)]I_i(t)/P(T > t) . \]

If input failure rates are constant in time, then the distribution of time to first occurrence of a basic event is exponential.

The occurrence rate of the top event is

\[ \text{all basic events} \sum \text{marginal occurrence rate} . \]

Again the formula for marginal occurrence rates holds only for trees not comprising NOT, EXOR, or exclusive \( x \) out of \( y \) gates.
1.4 The Program's Limitations

PAFT F77 is an exponential time process in terms of the number of replicated basic events contained in the inputed fault tree. Thus, the program is limited to trees with a few replicated basic events, approximately ten or less. The maximum dimension of an inputed tree is ninety-nine gates, including all basic events. Of these ninety-nine, a maximum of 20 replicated basic events can be specified, though a tree with this number will take days to compute. Replicated gates are not allowed.

A second limitation to the program is its inability to handle NO EXOR, or exclusive x out of y hates properly when calculating marginal occurrence rates. An enhancement to solve this problem can be readily coded into the state enumeration framework of the existing program.

An inadequate user interface is a third limitation to PAFT F77. An interface enhancement should prompt input either from an existing file or manual CRT entry. It should have limited editing capabilities in order to facilitate changes to existing input. And, idiot-proofing should be done to check for the correct data types and sizes of descriptive labels, probabilities between one and zero, and complete, connected, and acyclical fault trees.
2.1 Using PAFT F77 on UNIX

After having logged on successfully on the UNIX system, the PAFT F77 program can be run by giving the command,

\[ \text{ftree/paft/code} < \{\text{input file}\} > \{\text{output file}\} \]

\text{ftree/paft/code} specifies the location of the PAFT F77 machine code command file. The first parameter, preceded by \'<\' is the file name containing the data that defines the fault tree to be analysed. The second parameter preceded by \'\'> is optional. If given, output from the program is written into a file with the given name. The default is to have output directed back to one's terminal.

2.2 Input File Structure

The input file structure to PAFT F77 is a specific sequence. The following list indicates this order, with each number representing a line in the input file.

1. Fault tree name - any alpha-numeric entry up to 60 positions in length with no intermediate blanks.
2. Probabilities or failure rates - the character 'p' or 'f' is entered to specify the input as either in terms of probabilities or failure rates, respectively.

If failure rates are indicated, then lines 3 - 5 must be entered.

3. Number of time intervals - a digit is entered from 1 to a maximum of 9.
4 Time intervals - real nonnegative values are entered separated by blanks.

5 Time interval units - alpha-numeric entry up to 8 positions in length with no intermediate blanks.

6 Total number of nodes and number of replicated basic events - an integer from 1 to a maximum of 99, and an integer from 0 to a maximum of 20, respectively, separated by a blank.

7 Replicated node numbers (if any) - integer values separated by blanks.

Data 8 – 10 is entered for each node in the fault tree, starting with the top node as number 1.

8 Node description, node type, number of nodes above, number of nodes below - description is an alpha-numeric entry up to 16 positions in length with no intermediate blanks, type is an integer (see Table 1 for the index), number of nodes above and number of nodes below are integer entries from 0 to a maximum of 20. All entries on this line are separated by blanks.

If the node is a basic event, then line 9 must be entered.

9 Probability or failure rate - real nonnegative value.

10 Node number adjacencies - integer entries of node numbers adjacent from above then adjacent from below, separated by blanks.

Repeat 8 - 10 until all nodes have been defined.

The following two examples of input file structures for the two fault trees below should prove helpful.
TEST.PAFTCODE

OR gate #1

AND gate #2
Basic event #4
p = 0.1

AND gate #3
Basic event #5
p = 0.2

AND gate #6

Basic event #7
p = 0.5

Basic event #8
p = 0.6
A COMPLEX FAULT TREE EXAMPLE

Time intervals: 2.0, 3.5, 5.0 hours

Top
OR gate #1

AND gate #2

Basic event #20
fr = 0.001

Basic event #10
fr = 0.02

OR gate #4

Basic event #11
fr = 0.00004

OR gate #3

Basic event #12
fr = 0.0012

AND gate #5

Basic event #13
fr = 0.01

Basic event #14
fr = 0.9

AND gate #6

EXOR gate #15

Basic event #16
fr = 0.005

Basic event #21
fr = 0.073

Basic event #22
fr = 0.0032

Basic event #17
fr = 0.006

2/3 gate #7

Basic event #18
fr = 0.004

NOT gate #9

Basic event #19
fr = 0.01
Test.paftcode
p
8 1
5
top#or 3 0 2
2 3
gate#2and 2 1 2
1 4 5
gate#3and 2 1 2
1 5 6
be#4 1 1 0
0.1
2
be#5 1 2 0
0.2
2 3
gate#6and 2 1 2
3 7 8
be#7 1 1 0
0.5
6
be#8 1 1 0
0.6
6

INPUT FILE FOR: Test.paftcode
A.compex.fault.tree.example

f
3
2.0 3.5 5.0
hours
22 3
10 11 12
Top 3 0 3
2 3 11
Gate2 2 1 3
1 4 10 20
Gate3 3 1 2
1 5 11
Gate4 3 1 3
2 6 8 12
Gate5 2 1 3
3 12 13 14
Gate6 2 1 3
4 7 15 16
Gate7 6 1 3
6 9 17 18
Gate8 5 1 1
4 10
Gate9 5 1 1
7 19
RBE10 1 2 0
0.02
2 8
RBE11 1 2 0
0.00004
1 3
RBE12 1 2 0
0.0012

INPUT FILE FOR: A.compex.fault.tree.example
(continued)
INPUT FILE FOR: A.complex.fault.tree.example
TABLE 1

NODE TYPE INDEX

<table>
<thead>
<tr>
<th>Type index #</th>
<th>Gate type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Basic event</td>
</tr>
<tr>
<td>2</td>
<td>AND</td>
</tr>
<tr>
<td>3</td>
<td>OR</td>
</tr>
<tr>
<td>4</td>
<td>EXOR</td>
</tr>
<tr>
<td>5</td>
<td>NOT</td>
</tr>
<tr>
<td>6</td>
<td>2/3</td>
</tr>
<tr>
<td>7</td>
<td>x2/3</td>
</tr>
<tr>
<td>8</td>
<td>2/4</td>
</tr>
<tr>
<td>9</td>
<td>x2/4</td>
</tr>
<tr>
<td>10</td>
<td>3/4</td>
</tr>
<tr>
<td>11</td>
<td>x3/4</td>
</tr>
<tr>
<td>12</td>
<td>2/5</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>t</td>
<td>a/b</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>346</td>
<td>19/20</td>
</tr>
<tr>
<td>347</td>
<td>x19/20</td>
</tr>
</tbody>
</table>

\[ t = b^2 - 5b + 2a + 8 \]  \{adding 1 for exclusive cases\}
2.3 Output File Examples

The following two outputs are the results of running Test.paf/code and A.complex.fault.tree.example.

Note that in the second example the marginal occurrence rates of the basic events and the occurrence rates for the top event are invalid due to the presence of NOT gate #8, NOT gate #9 and EXOR gate #15.
Fault tree name: Test.paftcode  
Total number of events: 8  
Number of replicated events: 1

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Probability</th>
<th>Above# / Below#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>top#or</td>
<td>OR</td>
<td>0.740000e-01</td>
<td>/ 2 3</td>
</tr>
<tr>
<td>2</td>
<td>gate#2and</td>
<td>AND</td>
<td>0.200000e-01</td>
<td>1/ 4 5</td>
</tr>
<tr>
<td>3</td>
<td>gate#3and</td>
<td>AND</td>
<td>0.600000e-01</td>
<td>1/ 5 6</td>
</tr>
<tr>
<td>4</td>
<td>be#4</td>
<td>BE</td>
<td>0.100000e+00</td>
<td>2/</td>
</tr>
<tr>
<td>R- 5</td>
<td>be#5</td>
<td>BE</td>
<td>0.200000e+00</td>
<td>2 3/</td>
</tr>
<tr>
<td>6</td>
<td>gate#6and</td>
<td>AND</td>
<td>0.300000e+00</td>
<td>3/ 7 8</td>
</tr>
<tr>
<td>7</td>
<td>be#7</td>
<td>BE</td>
<td>0.500000e+00</td>
<td>6/</td>
</tr>
<tr>
<td>8</td>
<td>be#8</td>
<td>BE</td>
<td>0.600000e+00</td>
<td>6/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Probability</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>top#or</td>
<td>OR</td>
<td>0.740000e-01</td>
<td>0.100000e+01</td>
</tr>
<tr>
<td>R- 5</td>
<td>be#5</td>
<td>BE</td>
<td>0.200000e+00</td>
<td>0.370000e+00</td>
</tr>
<tr>
<td>4</td>
<td>be#4</td>
<td>BE</td>
<td>0.100000e+00</td>
<td>0.140000e+00</td>
</tr>
<tr>
<td>7</td>
<td>be#7</td>
<td>BE</td>
<td>0.500000e+00</td>
<td>0.108000e+00</td>
</tr>
<tr>
<td>8</td>
<td>be#8</td>
<td>BE</td>
<td>0.600000e+00</td>
<td>0.900000e+01</td>
</tr>
<tr>
<td>2</td>
<td>gate#2and</td>
<td>AND</td>
<td>0.200000e-01</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>gate#3and</td>
<td>AND</td>
<td>0.600000e-01</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>gate#6and</td>
<td>AND</td>
<td>0.300000e+00</td>
<td></td>
</tr>
</tbody>
</table>

OUTPUT FILE FOR: Test.paftcode
Fault tree name: A.complex.fault.tree.example
Total number of events: 22  Number of replicated events: 3
Time intervals: hours 2.00000 3.50000 5.00000

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Fail Rate</th>
<th>Above#</th>
<th>Below#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Top</td>
<td>OR</td>
<td></td>
<td>2/</td>
<td>3 11</td>
</tr>
<tr>
<td>2</td>
<td>Gate2</td>
<td>AND</td>
<td></td>
<td>1/</td>
<td>4 10 20</td>
</tr>
<tr>
<td>3</td>
<td>Gate3</td>
<td>OR</td>
<td></td>
<td>1/</td>
<td>5 11</td>
</tr>
<tr>
<td>4</td>
<td>Gate4</td>
<td>OR</td>
<td>2/</td>
<td>6 8 12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gate5</td>
<td>AND</td>
<td>3/</td>
<td>12 13 14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gate6</td>
<td>AND</td>
<td>4/</td>
<td>7 15 16</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gate7</td>
<td>2/3</td>
<td>6/</td>
<td>9 17 18</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Gate8</td>
<td>NOT</td>
<td></td>
<td>4/</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Gate9</td>
<td>NOT</td>
<td></td>
<td>7/</td>
<td>19</td>
</tr>
<tr>
<td>R-10</td>
<td>RBE10</td>
<td>BE</td>
<td>0.20000e-01</td>
<td>2/</td>
<td>8/</td>
</tr>
<tr>
<td>R-11</td>
<td>RBE11</td>
<td>BE</td>
<td>0.40000e-04</td>
<td>1/</td>
<td>3/</td>
</tr>
<tr>
<td>R-12</td>
<td>RBE12</td>
<td>BE</td>
<td>0.12000e-02</td>
<td>4/</td>
<td>5/</td>
</tr>
<tr>
<td>13</td>
<td>BE13</td>
<td>BE</td>
<td>0.10000e-01</td>
<td>5/</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>BE14</td>
<td>BE</td>
<td>0.90000e+00</td>
<td>5/</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Gate15</td>
<td>EXOR</td>
<td></td>
<td>6/</td>
<td>21 22</td>
</tr>
<tr>
<td>16</td>
<td>BE16</td>
<td>BE</td>
<td>0.50000e-02</td>
<td>6/</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>BE17</td>
<td>BE</td>
<td>0.60000e-02</td>
<td>7/</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>BE18</td>
<td>BE</td>
<td>0.40000e-02</td>
<td>7/</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>BE19</td>
<td>BE</td>
<td>0.10000e-01</td>
<td>9/</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>BE20</td>
<td>BE</td>
<td>0.10000e-02</td>
<td>2/</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>BE21</td>
<td>BE</td>
<td>0.73000e-01</td>
<td>15/</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>BE22</td>
<td>BE</td>
<td>0.32000e-02</td>
<td>15/</td>
<td></td>
</tr>
</tbody>
</table>

Time interval: 2.00000 hours
Occurrence rate of top event: 0.86746e-04

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Probability</th>
<th>Importance</th>
<th>Occur Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Top</td>
<td>OR</td>
<td>0.11980e-03</td>
<td>0.10000e+01</td>
<td>0.86746e-04</td>
</tr>
<tr>
<td>R-11</td>
<td>RBE11</td>
<td>BE</td>
<td>0.79997e-04</td>
<td>0.99996e+00</td>
<td>0.40000e-04</td>
</tr>
<tr>
<td>R-12</td>
<td>RBE12</td>
<td>BE</td>
<td>0.23971e-02</td>
<td>0.16604e-01</td>
<td>0.19879e-04</td>
</tr>
<tr>
<td>13</td>
<td>BE13</td>
<td>BE</td>
<td>0.19801e-01</td>
<td>0.20006e-02</td>
<td>0.19612e-04</td>
</tr>
<tr>
<td>20</td>
<td>BE20</td>
<td>BE</td>
<td>0.19980e-02</td>
<td>0.93493e-04</td>
<td>0.93317e-07</td>
</tr>
<tr>
<td>14</td>
<td>BE14</td>
<td>BE</td>
<td>0.83470e+00</td>
<td>0.47439e-04</td>
<td>0.70612e-05</td>
</tr>
<tr>
<td>R-10</td>
<td>RBE10</td>
<td>BE</td>
<td>0.39211e-01</td>
<td>0.47640e-05</td>
<td>0.91555e-07</td>
</tr>
<tr>
<td>17</td>
<td>BE17</td>
<td>BE</td>
<td>0.11928e-01</td>
<td>0.74371e-06</td>
<td>0.44095e-08</td>
</tr>
<tr>
<td>18</td>
<td>BE18</td>
<td>BE</td>
<td>0.79681e-02</td>
<td>0.74080e-06</td>
<td>0.29399e-08</td>
</tr>
<tr>
<td>16</td>
<td>BE16</td>
<td>BE</td>
<td>0.99502e-02</td>
<td>0.21311e-06</td>
<td>0.10551e-08</td>
</tr>
<tr>
<td>21</td>
<td>BE21</td>
<td>BE</td>
<td>0.13584e+00</td>
<td>0.14901e-07</td>
<td>0.94014e-09</td>
</tr>
<tr>
<td>22</td>
<td>BE22</td>
<td>BE</td>
<td>0.63796e-02</td>
<td>0.10994e-07</td>
<td>0.34961e-10</td>
</tr>
<tr>
<td>19</td>
<td>BE19</td>
<td>BE</td>
<td>0.19801e-01</td>
<td>-0.15069e-07</td>
<td>-0.14773e-09</td>
</tr>
<tr>
<td>2</td>
<td>Gate2</td>
<td>AND</td>
<td>0.18992e-06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Gate3</td>
<td>OR</td>
<td>0.11961e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Gate4</td>
<td>OR</td>
<td>0.96088e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gate5</td>
<td>AND</td>
<td>0.39620e-04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gate6</td>
<td>AND</td>
<td>0.27135e-04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gate7</td>
<td>2/3</td>
<td>0.19411e-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Gate8</td>
<td>NOT</td>
<td>0.96079e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Gate9</td>
<td>NOT</td>
<td>0.98002e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Gate15</td>
<td>EXOR</td>
<td>0.14049e+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OUTPUT FILE FOR: A.complex.fault.tree.example
### Time interval: 3.50000 hours

Occurrence rate of top event: 0.12454e-03

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Probability</th>
<th>Importance</th>
<th>Occur Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Top OR</td>
<td>OR</td>
<td>0.27894e-03</td>
<td>0.10000e+01</td>
<td>0.12454e-03</td>
</tr>
<tr>
<td>R-11</td>
<td>RBE11</td>
<td>BE</td>
<td>0.13999e-03</td>
<td>0.99986e+00</td>
<td>0.40000e-04</td>
</tr>
<tr>
<td>R-12</td>
<td>RBE12</td>
<td>BE</td>
<td>0.41912e-02</td>
<td>0.33144e-01</td>
<td>0.39618e-04</td>
</tr>
<tr>
<td>13</td>
<td>BE13</td>
<td>BE</td>
<td>0.34395e-01</td>
<td>0.40101e-02</td>
<td>0.38732e-04</td>
</tr>
<tr>
<td>20</td>
<td>BE20</td>
<td>BE</td>
<td>0.34939e-02</td>
<td>0.28297e-03</td>
<td>0.28206e-06</td>
</tr>
<tr>
<td>14</td>
<td>BE14</td>
<td>BE</td>
<td>0.95715e+00</td>
<td>0.14410e-03</td>
<td>0.55590e-05</td>
</tr>
<tr>
<td>R-10</td>
<td>RBE10</td>
<td>BE</td>
<td>0.67606e-01</td>
<td>0.14624e-04</td>
<td>0.27278e-06</td>
</tr>
<tr>
<td>17</td>
<td>BE17</td>
<td>BE</td>
<td>0.20781e-01</td>
<td>0.63009e-05</td>
<td>0.37030e-07</td>
</tr>
<tr>
<td>18</td>
<td>BE18</td>
<td>BE</td>
<td>0.13902e-01</td>
<td>0.62585e-05</td>
<td>0.24693e-07</td>
</tr>
<tr>
<td>16</td>
<td>BE16</td>
<td>BE</td>
<td>0.17348e-01</td>
<td>0.18094e-05</td>
<td>0.88926e-08</td>
</tr>
<tr>
<td>21</td>
<td>BE21</td>
<td>BE</td>
<td>0.22547e+00</td>
<td>0.13248e-06</td>
<td>0.74924e-08</td>
</tr>
<tr>
<td>22</td>
<td>BE22</td>
<td>BE</td>
<td>0.11138e-01</td>
<td>0.74386e-07</td>
<td>0.23545e-09</td>
</tr>
<tr>
<td>19</td>
<td>BE19</td>
<td>BE</td>
<td>0.34395e-01</td>
<td>-0.22557e-06</td>
<td>-0.21788e-08</td>
</tr>
<tr>
<td>2</td>
<td>Gate2</td>
<td>AND</td>
<td>0.10214e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Gate3</td>
<td>OR</td>
<td>0.27795e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Gate4</td>
<td>OR</td>
<td>0.93269e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gate5</td>
<td>AND</td>
<td>0.13798e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gate6</td>
<td>AND</td>
<td>0.13347e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gate7</td>
<td>2/3</td>
<td>0.33222e-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Gate8</td>
<td>NOT</td>
<td>0.93239e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Gate9</td>
<td>NOT</td>
<td>0.96561e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Gate15</td>
<td>EXOR</td>
<td>0.23159e+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Time interval: 5.00000 hours

Occurrence rate of top event: 0.15866e-03

<table>
<thead>
<tr>
<th>Node#</th>
<th>Description</th>
<th>Type</th>
<th>Probability</th>
<th>Importance</th>
<th>Occur Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Top OR</td>
<td>OR</td>
<td>0.49130e-03</td>
<td>0.10000e+01</td>
<td>0.15866e-03</td>
</tr>
<tr>
<td>R-11</td>
<td>RBE11</td>
<td>BE</td>
<td>0.19998e-03</td>
<td>0.99971e+00</td>
<td>0.40000e-04</td>
</tr>
<tr>
<td>R-12</td>
<td>RBE12</td>
<td>BE</td>
<td>0.59820e-02</td>
<td>0.48671e-01</td>
<td>0.58084e-04</td>
</tr>
<tr>
<td>13</td>
<td>BE13</td>
<td>BE</td>
<td>0.48771e-01</td>
<td>0.59116e-02</td>
<td>0.56260e-04</td>
</tr>
<tr>
<td>20</td>
<td>BE20</td>
<td>BE</td>
<td>0.49875e-02</td>
<td>0.57552e-03</td>
<td>0.57293e-06</td>
</tr>
<tr>
<td>14</td>
<td>BE14</td>
<td>BE</td>
<td>0.98889e+00</td>
<td>0.29155e-03</td>
<td>0.29164e-05</td>
</tr>
<tr>
<td>R-10</td>
<td>RBE10</td>
<td>BE</td>
<td>0.95163e-01</td>
<td>0.30163e-04</td>
<td>0.54612e-06</td>
</tr>
<tr>
<td>17</td>
<td>BE17</td>
<td>BE</td>
<td>0.29554e-01</td>
<td>0.23738e-04</td>
<td>0.13828e-06</td>
</tr>
<tr>
<td>18</td>
<td>BE18</td>
<td>BE</td>
<td>0.19801e-01</td>
<td>0.23514e-04</td>
<td>0.92238e-07</td>
</tr>
<tr>
<td>16</td>
<td>BE16</td>
<td>BE</td>
<td>0.24690e-01</td>
<td>0.68309e-05</td>
<td>0.33328e-07</td>
</tr>
<tr>
<td>21</td>
<td>BE21</td>
<td>BE</td>
<td>0.30580e+00</td>
<td>0.52343e-06</td>
<td>0.26538e-07</td>
</tr>
<tr>
<td>22</td>
<td>BE22</td>
<td>BE</td>
<td>0.15873e-01</td>
<td>0.20998e-06</td>
<td>0.66159e-09</td>
</tr>
<tr>
<td>19</td>
<td>BE19</td>
<td>BE</td>
<td>0.48771e-01</td>
<td>-0.12255e-05</td>
<td>-0.11663e-07</td>
</tr>
<tr>
<td>2</td>
<td>Gate2</td>
<td>AND</td>
<td>0.30079e-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Gate3</td>
<td>OR</td>
<td>0.48843e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Gate4</td>
<td>OR</td>
<td>0.90544e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gate5</td>
<td>AND</td>
<td>0.28831e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Gate6</td>
<td>AND</td>
<td>0.35756e-03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Gate7</td>
<td>2/3</td>
<td>0.46421e-01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Gate8</td>
<td>NOT</td>
<td>0.90484e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Gate9</td>
<td>NOT</td>
<td>0.95123e+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Gate15</td>
<td>EXOR</td>
<td>0.31197e+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OUTPUT FILE FOR: A.complex.fault.tree.example
3.1 Overview of Program Structure

PAFT F77 is comprised of fifteen Fortran F77 modules, in alphabetical order these are:

Subroutine bin(i) - adds 1 to the binary array bin(i) through a recursive procedure.

Function calpb(n) - calculates and returns the probability of node n.

Function calspb(n) - calculates and returns the probability of node n during state enumeration.

Subroutine import - calculates the marginal importance of all non-replicated basic events.

Subroutine input - reads the input file.

Program paft - the main module, performs subroutine calls and data initialization.

Subroutine match - sets flags for nodes dependent on replicated basic events.

Subroutine occur(ndone) - calculates the marginal occurrence rates of all basic events and the occurrence rate of the top event for the ndone-th time interval.

Subroutine output(ndone) - outputs data for a probability input, or for the ndone-th time interval of a failure rate input.

Subroutine qsort(m,n) - sorts the array S from subscript m to n; S indexes the marginal importances of basic events in decreasing order. Qsort uses a recursive procedure.

Subroutine rduce(i) - reduces the fault tree by calculating all gate probabilities not dependent on any replicated basic events. Rduce uses a recursive procedure starting with node 1.
Subroutine `remark(i)` - temporarily resets the flags signifying nodal dependence on replicated basic events, for the nodes above non-replicated basic event \( i \).

Subroutine `sreduce(i)` - reduces the fault tree by calculating all gate probabilities during a specific state enumeration. `sreduce` uses a recursive procedure starting with node 1.

Subroutine `states` - calculates all gate probabilities dependent on replicated basic events through the state enumeration method. Also calculates the marginal importances of all replicated basic events.

Function `wrdtyp(i)` - a character function returning the alphanumeric equivalent of the type index \( i \).

The diagram below shows the structure of the procedural dependence between the modules.

```
  wrdtyp  occurr  qsort  calpb
    /   |     |     |
  |     |     |     |
  output  rduce

  input  paft  match
    |     |
  |     |
  |     |
  import  bin  states
    |     |
  |     |
  |     |
  remark  sreduce
    |
  calspb
```
Two common data blocks are used to define global variables; nn, for numerical data, and cc, for character data.

3.2 Global Variable Definitions

The following list gives the global variable definitions for PAFT F77. They are ordered according to their position in the common blocks cc and nn.

Common block cc -

name - the fault tree name; up to 60 characters.
descrp(99) - node descriptions; up to 16 characters each.
units - time units for failure rate input; up to 8 characters.
pf - probability or failure rate input indicator; 1 character.

Common block nn -

type(99) - node type indexes; integer.
above(99) - number of nodes above; integer.
below(99) - number of nodes below; integer.
nodes(99,20) - adjacent node numbers above and then below; integer.
prb(99) - probability of nodes; real.
sprb(99) - scratch probability of nodes during state enumeration; real.
prt(99) - marginal importance of nodes; real.
fail(99) - basic event failure rates; real.
time(9) - time intervals; real.
sumocr - occurrence rate of top event; real.
ocr(99) - marginal occurrence rates of basic events; real.
total - total number of nodes; integer.
nrep - number of replicated basic events; integer.

nbe - number of basic events; integer.

replc(20) - replicated basic event node numbers; integer.

numti - number of time intervals; integer.

mark(99) - flag signifying nodal dependence on a replicated basic event or an enumerated basic event; integer.

store(99) - scratch store for mark(99); integer

bn(20) - binary array signifying states of replicated basic events; integer.

dpend(99) - flag signifying nodal dependence on a replicated basic event through two or more paths; integer.

s(99) - index for order of marginal importances; integer.
APPENDIX

subroutine bin(i)
integer bn(20)
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
&          ptt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
&          replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
change i th member of bn, 0 to 1 or 1 to 0
if (bn(i).eq.0) then
  bn(i)=1
else
  bn(i)=0
end if
if 0 to 1 return
  return
else
if 1 to 0 change i+1 th member of bn
  call bin(i+1)
end if
return
end

function calpb(n)
real p,pl
integer a,b,c,d,fin,start,num,bn(20),type(99),above(99),below(99),
&          nodes(99,20)
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
&          ptt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
&          replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
a=above(n)
b=below(n)
type of gate defines calpb operation
if (type(n).ge.6) go to 50
  p=prb(nodes(n,a+1))
c=2
  go to (10,20,30,40) type(n)-1
AND gate
10  do 1 i=a+2, a+b
      p=p*prb(nodes(n,i))
  1  continue
  go to 70
OR gate
20  c=1
EXOR gate
30  do 2 j=a+2, a+b
      pl=prb(nodes(n,j))
      p=p+pl-c*p*pl
  2  continue
  go to 70
NOT gate
40  p=1-p
  go to 70
all other gates
50  p=0.0
change type index to (x)a/b form
d=(type(n)-b*b+5*b-8)/2
bn(0)=0
continue
set up correct iteration bounds for generating function operations
if (mod(type(n),2).eq.1) then
    start=d
    fin=d
    else
        start=0
        fin=d-1
        else
            start=d
            fin=b
        end if
    end if
generating function operation
do 4 k=1, b**2
    num=0
    do 5 l=1, b
        if (bn(l).eq.1) num=num+1
        continue
        if ((num.lt.start).or.(num.gt.fin)) go to 60
        pl=1.0
    end if
probability calculation
    do 6 m=1, b
        if (bn(m).eq.1) then
            pl=pl*prb(nodes(n,a+m))
        else
            pl=pl*(1-prb(nodes(n,a+m)))
        end if
        continue
    p=p+pl
60 if (k.lt.b**2) call bin(1)
4 continue
if (d.le.b/2) p=1-p
70 calpb=p
return
end

function calspb(n)
real p,pl
integer a,b,c,d,fin,start,num,bn(20),type(99),above(99),below(99),
& nodes(99,20)
& common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
& prc(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
& replc(20),numti,mark(99),store(99),bn(20),dpend(99),a(99)
& a=above(n)
b=below(n)
c type of gate defines calspb operation
if (type(n).ge.6) go to 50
p=sprb(nodes(n,a+1))
c=2
  go to (10,20,30,40) type(n)-1
  c AND gate
10  do 1 i=a+2, a+b
    p=p*sprb(nodes(n,i))
  1  continue
  go to 70
  c OR gate
20  c=1
  c EXOR gate
30  do 2 j=a+2, a+b
    pl=sprb(nodes(n,j))
    p=p+pl-c*p*pl
  2  continue
  go to 70
  c NOT gate
40  p=1-p
  go to 70
  c all other gates
50  p=0.0
  c change type index to (x)a/b form
d=(type(n)-b*b+5*b-8)/2
  do 3 g=1, b
    bn(g)=0
  3  continue
  c set up correct iteration bounds for generating function operations
if (mod(type(n),2).eq.1) then
  start=d
  fin=d
  else
    if (d.1e,b/2) then
      start=0
      fin=d-1
      else
      start=d
      fin=b
      end if
      c generating function operation
    end if
  do 4 k=1, b**2
    num=0
  do 5 l=1, b
    if (bn(l).eq.1) num=num+1
  5  continue
    if ((num.lt.start).or.(num.gt.fin)) go to 60
    pl=1.0
  c sprob calculations
  do 6 m=1, b
if (bn(m).eq.1) then
    pl=pl*sprb(nodes(n,a+m))
else
    pl=pl*(1-sprb(nodes(n,a+m)))
end if

continue
p=p+pl

if (k.lt.b**2) call bin(l)
continue
if (d.le.b/2) p=1-p
calspb=p
return
end

subroutine import
integer l,c,type(99),above(99),total,nrep,replc(20),bn(20),
   & store(99),mark(99)
real p,x
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
   & prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
   & replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
c top event always has a marginal importance of 1.0
prl(1)=1.0
c reclaim original mark flags
do 1 i1l, total
   store(i)=mark(i)
1 continue
c select only nonreplicated basic events
do 2 n1l, total
   if (type(n).gt.l) go to 2
   if (above(n).gt.l) go to 2
   call remark(n)
sprb(n)=0.0
c basic event first off then on
do 3 m=l, 2
   if (m.eq.2) sprb(n)=1.0
   do 4 g=l, nrep
      bn(g)=0
4 continue
c=2**nrep
c state enumeration on nonreplicated basic event
do 5 j=l,c
   p=1.0
   do 6 k=l, nrep
      l=replc(k)
sprb(l)=bn(k)
6 continue
c probability calculation
   if (bn(k).eq.1) then
      p=p*prb(l)
   else
      p=p*(1-prb(l))
   end if
continue
call srduce(1)
x*p*sprb(1)
c marginal importance calculation
if (m.eq.2) then
    prt(n)=prt(n)+x
else
    prt(n)=prt(n)-x
end if
if (j.lt.c) call bin(1)
continue
continue
do 7 h=1, total
    if (store(h).eq.l) sprb(h)=prb(h)
continue
2 continue
return
end

subroutine input
integer type(99),above(99),below(99),nodes(99,20),total,numti,
& nrep,replc(20)
character name*60,descrp(99)*16,units*8,pf
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
& prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
& replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
common /cc/ name,descrp(99),units,pf
read(5,*) name
read(5,*)
pf
if (pf.eq.'p') go to 10
read(5,*) numti
read(5,*) (time(k),k-l, numti)
read(5,*) units
10 read(5,*) total,nrep
if (nrep.gt.0) read(5,*) (replc(n),n-l, nrep)
do 1 i=1, total
    read(5,*) descrp(i),type(i),above(i),below(i)
    if (type(i).eq.1) then
        read(5,*) prb(i)
c fail to be changed to a probability in main routine
    fail(i)=prb(i)
    end if
    read(5,*) (nodes(i,j),j-l, above(i)+below(i))
1 continue
return
end

program paft
integer j,n,type(99),total,nrep,above(99),below(99),replc(20),numti,
& nbe,nodes(99,20),mark(99),s(99),bn(20),dpend(99),store(99)
real prb(99),sprb(99),prt(99),fail(99),time(9),sumocr,
& ocr(99)
CALL INPUT
C PR(TOP) IS ALWAYS EQUAL TO 1.0
S(1)=1
NBE=1
C J KEEPS COUNT OF NON-BASIC EVENTS
J=TOTAL
C SET UP S ARRAY FOR CORRECT INPUT TO QSORT AND OUTPUT Routines
DO 1 K=1, TOTAL
   DPEND(K)=0
   MARK(K)=1
   IF (K.EQ.1) GO TO 1
   N=TOTAL+2-K
   IF (TYPE(N).EQ.1) THEN
      NBE=NBE+1
      S(NBE)=N
   ELSE
      S(J)=N
      J=J-1
   END IF
1 CONTINUE
CALL MATCH
C NUMBER OF OUTPUT LOOPS
IF (PF.EQ.'P') NUMTI=1
DO 2 NDONE=1, NUMTI
C INITIALIZATIONS; FAIL INTO PRB, PRB INTO SPRB, AND PRT
DO 3 I=1, TOTAL
   IF (TYPE(I).EQ.1) THEN
      IF (PF.EQ.'F') PRB(I)=1-EXP(-FAIL(I)*TIME(NDONE))
      ELSE
      PRB(I)=0.0
   END IF
   SPRB(I)=PRB(I)
   PRT(I)=0
3 CONTINUE
CALL REDUCE(1)
IF (NREP.EQ.0) GO TO 10
CALL STATES
10 CALL IMPORT
CALL OUTPUT(NDONE)
2 CONTINUE
STOP
END

SUBROUTINE MATCH
INTEGER L,C,N,J,List(99),Above(99),Nodes(99,20),Dpend(99),Nrep,
&       MARK(99),Replc(20)
do 1 i=1, nrep
  list stores replicated basic events already processed
  list(l)=replc(i)
  l=1
  c=1
  20 do 2 k=1, above(list(c))
  c process starts at replicated basic event and marks up the fault tree
     n=nodes(list(c),k)
     j=1
     if(list(j).eq.n) dpend(n)=l
     if (dpend(n).eq.1) go to 2
     j=j+1
     if (1.1e.1) go to 10
     1=1+l
     list(l)=n
     mark(n)=0
  2 continue
  c next replicated basic event
     c=c+1
     if (c.le.l) go to 20
  1 continue
  return
end

subroutine occur(ndone)
  integer type(99),total
  real q
  common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
  & prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
  & replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
  sumocr=0.0
  q=1-prb(1)
  do 1 k=2, total
     if (type(k).eq.1) then
       ocr(k)=(1-prb(k))*fail(k)*prt(k)/q
     sumocr=sumocr+ocr(k)
     end if
  1 continue
  ocr(1)=sumocr
  return
end

subroutine output(ndone)
  integer a,b,type(99),above(99),below(99),nodes(99,20),total,numti,
  & nrep,s(99),nbe
  character name*60,descrip(99)*16,blanks*11,units*8,word*6,wrdtype*6,r*2,pf
  common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
  & prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
  & replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
common /cc/ name, descrp(99), units, pf
blanks ='

c go to time interval output code if ndone is greater than 1
if (ndone.gt.1) go to 1000
open(6, form='print')
write(6, 10) name

10 format('OFault tree name: ', a60)
write(6, 20) total.nrep

20 format(' Total number of events: ', i2, ' Number of replicated',
& ' events: ', i2)

c go to failure rate output code if input is in terms of failure rates
if (pf.eq.'f') go to 2000
c probability's initial output
write(6, 30)

30 format('ONode# Description Type Probability ',
& 'Above# / Below#')
do 1 k=1, total
a = above(k)
b = below(k)
r =
if (a.gt.l) r = 'R-
word = wrdtype(type(k))
write(6, 40) r, k, descrp(k), word, prb(k)
40 format(1x, a2, i3, 3x, al6, 1x, a4, 2x, el2.5, $)
if (a.eq.0) write(6, 50) blanks
if (a.eq.1) write(6, 60) nodes(k, l)
if (a.eq.2) write(6, 70) nodes(k, l), nodes(k, 2)
if (a.gt.2) write(6, 80) (nodes(k, i), i = 1, a)
50 format(1x, al$, $)
60 format(9x, i3, $)
70 format(6x, 2i3, $)
80 format(3x, 20(i3, $))
write(6, 90) (nodes(k, j), j = a+1, a+b)
90 format(1x, ' /', 2o3)
1 continue

c probability's secondary output
write(6, 100)
100 format(' Marginal')
write(6, 110)

110 format(' Node# Description Type Probability Importance')
call qsort(2, nbe)
do 2 n = 1, total
r =
if (above(s(n)).gt.1) r = 'R-
word = wrdtype(type(s(n)))
if (((n, eq.1).or.(type(s(n)).eq.1)) then
write(6, 120) r, s(n), descrp(s(n)), word, prb(s(n)), prt(s(n))
else
write(6, 120) r, s(n), descrp(s(n)), word, prb(s(n))
end if
120 format(1x, a2, 13, 3x, al6, 1x, a4, 2x, el2.5, 2x, el2.5, 2x, el2.5)
continue
return
c failure rate's initial output
2000 write(6,130) units,(time(f),f=1, numti)
130 format(' Time intervals: ',a8,3f8.5)
write(6,140)
140 format('ONode# Description Type Fail Rate Above#',
& ' / below#')
do 3 l=1, total
a=above(l)
b=below(l)
r=
if (a.gt.1) r='R-'
word=wrdtype(type(l))
if (type(l).eq.1) then
write(6,150) rl,descrp(l),word,fail(l)
else
write(6,160) rl,descrp(l),word,'end if
150 format(lx,a2,i3,3x,al6,lx,a4,2x,e12,5,$)
160 format(lx,a2,i3,3x,al6,lx,a14,2x,e12,5,$)
if (a.eq.0) write(6,50) blanks
if (a.eq.1) write(6,60) nodes(l,1)
if (a.eq.2) write(6,70) nodes(l,1),nodes(l,2)
if (a.gt.2) write(6,80) (nodes(l,m),m-l, a)
write(6,90) (nodes(l,g),g-a+l, a+b)
3 continue
c failure rate's secondary output
1000 write(6,170) time(ndone),units
170 format('Time interval: ',f8.5,lx,a8)
call occur(ndone)
write(6,180) sumocr
180 format(' occurrence rate of top event: ',e12.5)
write(6,190)
190 format('Marginal ',
& 'Marginal')
write(6,200)
200 format('Node# Description Type Probability Importance',
& 'Occur Rate')
call qsort(2,nbe)
do 4 h=1, total
r=
if (above(s(h)).gt.1) r='R-'
word=wrdtype(type(s(h)))
if ((h.eq.1).or.(type(s(h)).eq.1)) then
write(6,120) r,s(h),descrp(s(h)),word,prb(s(h)),prt(s(h)),ocr(s(h))
else
write(6,120) r,s(h),descrp(s(h)),word,prb(s(h))
end if
4 continue
return
subroutine qsort(m,n)
  integer i(99),ii(99),j(99),jj(99),l,k,s(99),total
  real x
  common /nn/ type(99),above(99),below(99),nodes(99,20),r-b(99),sprb(99),
  &    prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
  &    replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
  if ((m.eq.2).and.(n.eq.total)) l=0
  c 1 is recursive level indicator
  l=l+1
  ii(l)=m
  jj(l)=n
  i(l)=m
  j(l)=n
  x=prt(s(j(1)))
  c standard recursive quicksort routine
  10 if (prt(s(i(1))).le.x) go to 20
     i(l)=i(l)+1
     go to 10
  20 if (prt(s(j(1))).ge.x) go to 30
     j(l)=j(l)-1
     go to 20
  30 if (i(l).le.j(l)) then
     k=s(i(l))
     s(i(l))=s(j(l))
     s(j(l))=k
     i(l)=i(l)+1
     j(l)=j(l)-1
     end if
     l=l-1
    c resort all smaller than x
     if (ii(l).lt.j(l)) then
         call qsort(ii(l),j(l))
         l=l-1
         end if
    c resort all larger than x
     if (i(l).lt.jj(l)) then
         call qsort(i(l),jj(l))
         l=l-1
         end if
    return
end

subroutine rduce(i)
  integer n(99),k(99),l,above(99),below(99),nodes(99,20),dpend(99)
  common /nn/ type(99),above(99),below(99),nodes(99,20),r-b(99),sprb(99),
  &    prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
  &    replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
  if (i.eq.1) then
    l=0
    n(l)=1
    end if
    c 1 is recursive level indicator
    l=l+1
c k is iterative breadth pointer
k(1)=above(n(1))+1
10 n(1+l)=nodes(n(1),k(1))
c recurse depth first
if (below(n(1+l))).gt.0) call rduce(n(1+l))
iterate breadth second
if (k(1).lt.above(n(1))+below(n(1))) then
  k(1)=k(1)+1
to to 10
end if
c probability calculations
if (dpend(n(1)).eq.0) then
  prb(n(1))=calpb(n(1))
  sprb(n(1))=prb(n(1))
end if
l=1-1
returnend

subroutine remark(i)
integer j,n,store(99),nodes(99,20),mark(99),total
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
replic(20),numti,mark(99),store(99),bn(99),dpend(99),s(99)
c store original mark in store array
do 1 k=1, total
  mark(k)=store(k)
1 continue
j=i
10 n=nodes(j,1)
c unmark all nodes above i
  mark(n)=0
  j=n
if (j.gt.1) go to 10
return
end
subroutine srduce(i)
integer n(99),k(99),l,above(99),below(99),nodes(99,20),mark(99)
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprb(99),
prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
replic(20),numti,mark(99),store(99),bn(99),dpend(99),s(99)
if (i.eq.1) then
  l=0
  n(1)=1
end if
c l is recursive level indicator
l=l+1
c k is iterative breadth pointer
k(1)=above(n(1))+1
10 n(1+l)=nodes(n(1),k(1))
c recurse depth first
if (mark(n(1+l)).eq.0) call rduce(n(1+l))
c iterate breadth second
if (k(l).lt.above(n(l))+below(n(l))) then
  k(l)=k(l)+1
  go to 10
end if

c sprob calculations
sprob(n(l))=calspb(n(l))
l=l-1
returnend subroutine states

integer l,c,type(99),total,nrep,replc(20),bn(20),dpend(99)
real p,x
common /nn/ type(99),above(99),below(99),nodes(99,20),prb(99),sprob(99),
& prt(99),fail(99),time(9),sumocr,ocr(99),total,nrep,nbe,
& replc(20),numti,mark(99),store(99),bn(20),dpend(99),s(99)
do 1 i=1, nrep
  bn(i)=0
  continue
state enumeration process
c=2**nrep
do 2 j=1, c
  p=1.0
  do 3 k=1, nrep
    l=replc(k)
    sprb(l)=bn(k)
  end do
  c probability calculations
  if (bn(k).eq.1) then
    p=p*prb(l)
  else
    p=p*(1-prb(l))
  end if
  continue
  call srduce(l)
do 4 n=1, total
    if (dpend(n).eq.0) go to 4
    x=p*sprb(n)
    prb(n)=prb(n)+x
    if (n.gt.1) go to 4
  end do
  replicated basic event marginal importances calculations
  do 5 m=1, nrep
    l=replc(m)
    if (bn(m).eq.1) then
      prt(l)=prt(l)+x/prb(l)
    else
      prt(l)=prt(l)-x/(1-prb(l))
    end if
    continue
  end do
  4 continue
  if (j.lt.c) call bin(l)
2 continue
returnend
character*6 function wrdtyp(i)
integer a,b,ind
character word(5)*6, cha*2, chb*2
data word/'BE','AND','OR','EXOR','NOT'/
c if type index is less than 6 return correct data word
if (i.lt.6) then
  wrdtyp=word(i)
  return
end if
c change type index to (a)a/b alpha-numeric form
b=(sqrt(4*i-23.)+5)/2
a=(i-b*b+5*b-8)/2
write(cha,'(i2)') a
write(chb,'(i2)') b
if (mod(i,2).ne.0) then
  wrdtyp='X'
  ind=2
else
  ind=1
end if
if (a.ge.10) then
  wrdtyp(ind:)=cha//'I'
  ind=ind+3
else
  wrdtyp(ind:)=cha(2:2)//'I'
  ind=ind+2
end if
if (b.ge.10) then
  wrdtyp(ind:)=chb
  ind=ind+2
else
  wrdtyp(ind:)=chb(2:2)
  ind=ind+1
end if
return
end
REFERENCES

