A Verification Study on the Loop-Breaking Logic of FTREX

Jong-Soo Choi
Korea Institute of Nuclear Safety, 19 Guseong-dong Yuseong-gu Daejeon 305-338, k209cjs@kins.re.kr

1. Introduction

The logical loop problem in fault tree analysis (FTA) has been solved by manually or automatically breaking their circular logics. The breaking of logical loops is one of uncertainty sources in fault tree analyses. A practical method which can verify fault tree analysis results was developed by Choi [1]. The method has the capability to handle logical loop problems. It has been implemented in a FORTRAN program which is called VETA (Verification and Evaluation of fault Tree Analysis results) code.

FTREX [2], a well-known fault tree quantifier developed by KAERI, has an automatic loop-breaking logic. In order to make certain of the correctness of the loop-breaking logic of FTREX, some typical trees with complex loops are developed and applied to this study. This paper presents some verification results of the loop-breaking logic tested by the VETA code.

2. Methods and Results

2.1 Method for Verifying FTA Results

In quantifying a fault tree including a looping tree, a set of minimal cut sets (MCSs) leading to the top event are developed. All the developed MCSs meet the Boolean equations of the top event. If the system state is equivalent to one of the developed MCSs, the top event must occur. Particularly for logical loop problems, we can verify fault tree analysis results as shown in Fig. 1.

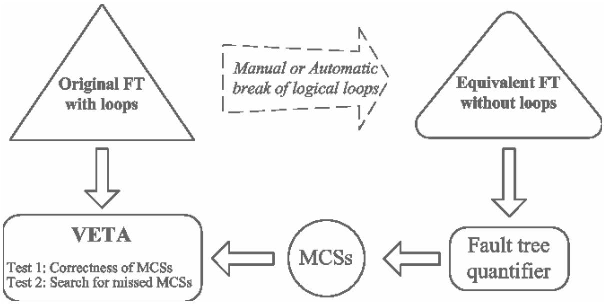
Original FT with loops
Manual or Automatic break of logical loops
Equivalent FT without loops
VETA
Test 1: Correctness of MCSs
Test 2: Search for missed MCSs
MCSs
Fault tree quantifier

Figure 1. Verification method for logical loop problems

Every MCS inherently has no proper subset as a cut set. Thus each developed MCS **K** should meet the following properties:

<u>Property 1</u>: If $\mathbf{B_1}(\mathbf{X}) = \mathbf{K}$, then $\phi(\mathbf{X}) = 1$.

<u>Property 2</u>: If $\mathbf{B_1}(\mathbf{X}) = \mathbf{K}$ and $i \in \mathbf{K}$, then $\phi(0_i,\mathbf{X}) = 0$.

Using these properties, we can test the correctness of the developed MCSs. If any MCS does not meet all these properties, we can conclude that there are some drawbacks to the fault tree quantifier used to generate the MCSs. (Correctness Test of MCSs)

Similarly to the truncation error evaluation using Delta-X Monte Carlo method [3], we can find the unidentified cut sets which meet $\delta(\mathbf{X}) = 1$ by Monte Carlo simulations with large sample size. If we get a new unidentified cut set, we also obtain its corresponding MCS by the recursive calculations of $\phi(0_i,\mathbf{X})$ for $i \in \mathbf{B_1}(\mathbf{X})$. Consequently, new MCSs will be developed. If any missed MCS is identified, we can conclude that there are some drawbacks to the fault tree quantifier used to generate the MCSs. (Search for Missed MCSs)

The verification method of VETA, which is based on characteristics of coherent reliability systems and Monte Carlo method, is simple and flexible. So, this method can be easily used to verify analysis results of any fault trees including looping trees.

2.2 Development of Example Looping Trees

The capital letters *A, B, C, …* indicate gates and the small letters *a, b, c, …* are basic events. The top event of a fault tree can be written as:

$$TOP = f(A,B,\Lambda\ ,X_1,X_2,\Lambda\ ,Y_1,Y_2,\Lambda\ ,a,b,\Lambda\ ) \qquad (1)$$

where X_i is the loop-causing gate for the *i*-th loop and Y_i is the feedback gate for X_i. All the gates and the basic events in Eq. (1) are relevant to the top events. Each gate can be written as:

$$G := (A,B,\Lambda\ ,a,b,\Lambda\ ) \qquad (2)$$

where *A, B, …, a, b, …* are the input events of the output gate *G*.

In order to test the loop-breaking logic of FTREX, the following 5 looping trees are made by modifying the tree 'European 1' (given in Ref. [4]).

<u>CASE 1</u>: single loop
In this case, the Boolean equation of the loop-cause gate *X* and the feedback gate *Y* can be defined as:
$X = f(Y, \ldots)$, $Y := (X, \ldots)$
An example tree of this case is made by modifying the tree 'European 1'
from: G068 := (C001 & C008)
to: G068 := (C001 & C008 & ROOT).
Then X = ROOT, Y = G068.

<u>CASE 2</u>: Inner loop
$X_1 = f(X_2, Y_2, Y_1, \ldots)$, $Y_1 := (X_1, \ldots)$
$X_2 = f(Y_2, \ldots)$, $Y_2 := (X_2, \ldots)$
An example tree of this case is made by modifying
from: G068 := (C001 & C008)
G128 := (G116 & G120)
to: G068 := (C001 & C008 & ROOT)

G128 := (G116 & G120 & G143). Then
X_1 = ROOT, Y_1 = G068, X_2 = G143, Y_2 = G128.

CASE 3: Two independent loops
$X_1 = f(Y_1, \ldots)$, $Y_1 := (X_1, \ldots)$
$X_2 = f(Y_2, \ldots)$, $Y_2 := (X_2, \ldots)$
An example tree of this case is made by modifying
from: G068 := (C001 & C008)
G063 := (C001 & C003)
to: G068 := (C001 & C008 & G140)
G063 := (C001 & C003 & G139). Then
X_1 = G140, Y_1 = G068, X_2 = G139, Y_2 = G063.

CASE 4: Multi-feedback gates
$X = f(Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7, Y_8, \ldots)$
$Y_1 = f(X, \ldots), \ldots, Y_8 := (X, \ldots)$
An example tree of this case is made by modifying
from: G069 := (C001 & C009)
G068 := (C001 & C008)
…
G062 := (C001 & C002)
to: G069 := (C001 & C009 & ROOT),
G068 := (C001 & C008 & ROOT)
…
G062 := (C001 & C002 & ROOT). Then
X = ROOT, Y_1 = G069, Y_2 = G068, ..., Y_8 = G062.

CASE 5: Two loops with single feedback gate
$X_1 = f(Y, \ldots)$, $X_2 = f(Y, \ldots)$, $Y := (X_1, X_2, \ldots)$
An example tree of this case is made by modifying
from: G065 := (C001 & C005)
to: G065 := (C001 & C005 & G141 & G138). Then
X_1 = G141, X_2 = G138, Y = G065.

2.3 Fault Tree Analysis

Using an analytical method [2], FTREX breaks the logical loops in merged fault tree by disconnecting one of the connected gates that cause the logical loop and then solve the broken fault tree. FTREX could solve large coherent fault tree with a small memory usage in a short time.

Table 1 shows that FTREX solves the 5 looping tree described in Sec. 2.2 without cut-off.

Table 1: Quantification results calculated by FTREX

Looping Trees	# of MCSs (without cut-off)	Probability of top event
Case 1	32876	7.1306E-7
Case 2	27750	7.1251E-7
Case 3	62175	9.0389E-7
Case 4	2784	4.6636E-8
Case 5	53320	7.7090E-7

2.4 Verification of FTREX's loop-breaking logic

VETA, a verification tool of FTA results, has two testing modules:

1) Correctness Test of MCSs
2) Search for Missed MCSs

If any FTA result for any example tree fails in these tests, we can conclude that there are some drawbacks to the fault tree quantifier used to generate the MCSs.

Table 2 shows VETA results of the developed MCSs described in Table 1 with sample size of 10^{12}. Consequently, no drawbacks to the loop-breaking logic of FTREX are found from the 5 typical example looping problems.

Table 2: Verification results calculated by VETA

Looping Trees	Correctness Test of MCSs	Search for Missed MCSs*
Case 1	OK (all MCSs)	Not found
Case 2	OK (all MCSs)	Not found
Case 3	OK (all MCSs)	Not found
Case 4	OK (all MCSs)	Not found
Case 5	OK (all MCSs)	Not found

* Sample size of Monte Carlo simulation: 1E12

3. Conclusion

This paper presents the verification results of the loop-breaking logic of FTREX, which are obtained from VETA calculations for the five typical looping trees. Judging from this study, FTREX exactly solves coherent looping-tree problems.

REFERENCES

[1] J. S. Choi, Verification of Fault Tree Analysis Results Particularly in the Logical Loop Problems, Transaction of KNS Autumn Meeting, Pyeong-Chang, Oct. 25-26, p.535, 2007.
[2] W. S. Jung and S. H. Han, Development of an Analytical Method to Break Logical Loops at the System Level, Reliability Engineering and System Safety, Vol.90, p.37, 2005.
[3] J. S. Choi and N. Z. Cho, A Practical Method for Accurate Quantification of Large Fault Trees, Reliability Engineering and System Safety, Vol.92, p.971, 2007.
[4] A. Rauzy, New Algorithms for Fault Trees Analysis, Reliability Engineering and System Safety, Vol.40, p.203, 1993.