Application of Shannon Decomposition to Event Tree Quantification

Jongsoo Choi
Korea Institute of Nuclear Safety, 62 Gwahak-ro, Yuseong-gu, Daejeon 34142
Corresponding author: k209cjs@kins.re.kr

1. Introduction

The fault tree/event tree methodology is widely used in the nuclear industry to obtain risk models for probabilistic safety assessment (PSA) studies. The classical methodology to assess these models is based on the computation of the minimal cutsets (MCS). The MCSs are essentially applicable for coherent fault trees.

For many real PSA models, the full conversion procedure remains out of reach in terms of computational resources owing to their size, non-coherency, and complexity. A potential solution to improve the quality of assessment methods is to design hybrid algorithms that combine the information derived from the calculation of MCS with the Binary Decision Diagrams (BDDs) methodology [1,2,3].

BDD is a popular data structure for representation and manipulation of Boolean functions. It is based on the well-known Shannon decomposition, or Boole's expansion theorem, also called pivotal decomposition.

In this paper, an implementation of the Shannon decomposition method is proposed for complex and non-coherent event tree quantification.

2. Shannon Decomposition

We extend the Shannon decomposition of Boolean functions to more general classes of functions. A fault tree is used to represent a binary system. A binary system is a system in which all components and the entire system are assumed to be either completely operational or completely failed. A binary system is denoted F where X is the set of components and F is a binary function of the component states.

The Shannon decomposition is the following identity:

$$F = x \cdot F_x + x' \cdot F_{x'} \quad (1)$$

where F is any Boolean function of coherent or non-coherent fault tree, x is a variable, x' is the complement of x, and F_x and $F_{x'}$, are F with the argument x set equal to 1 and to 0 respectively. The terms F_x and $F_{x'}$, are sometimes called the positive and negative Shannon cofactors, respectively, of F with respect to x.

A more explicit way of stating the Shannon decomposition is:

$$F(x_1, x_2, \ldots, x_n) = x_1 \cdot F(1, x_2, \ldots, x_n) + x_1' \cdot F(0, x_2, \ldots, x_n). \quad (2)$$

In Exclusive OR (XOR) form, Eq. (2) also holds when the disjunction "+" is replace by the XOR operator $\oplus$:

$$F(x_1, x_2, \ldots, x_n) = x_1 \cdot F(1, x_2, \ldots, x_n) \oplus x_1' \cdot F(0, x_2, \ldots, x_n). \quad (3)$$

Repeated application for each argument leads to the Sum of Products (SOP) canonical form of the function F. For $n = 2$ that would be

$$F(x_1, x_2) = x_1 \cdot F(1, x_2) + x_1' \cdot F(0, x_2) = x_1 x_2 \cdot F(1, 1) + x_1 x_2' \cdot F(1, 0) + x_1' x_2 \cdot F(0, 1) + x_1' x_2' \cdot F(0, 0). \quad (4)$$

By N/2 decompositions respect to arbitrarily selected variables, we can get N decomposed functions which are mutually exclusive. The Boolean function F of the original fault tree can be expressed in terms of the decomposed trees and their conditional events as

$$F(\mathrm{X}) = \mathrm{X}_1 \cdot F_1 \oplus \mathrm{X}_2 \cdot F_2 \oplus \ \ldots \ \oplus \mathrm{X}_N \cdot F_N \quad (5)$$

where

$F(\mathrm{X})$: original fault tree

X_i : i-th condition event of which argument is set equal to 1 or 0

F_i : conditional event of $F(\mathrm{X})$ given the i-th condition event $(F_i \mid \mathrm{X}_i)$

Because the union of all the condition events is the universal set and each condition event is mutually exclusive, then

$$\Pr(\mathrm{X}_1) + \Pr(\mathrm{X}_2) + \ldots + \Pr(\mathrm{X}_N) = 1. \quad (6)$$

From Eq. (5), the probability of the top event of an original fault tree can be calculated as follows:

$$\Pr[F(\mathrm{X})] = \Pr(\mathrm{X}_1)\Pr(F_1) + \Pr(\mathrm{X}_2)\Pr(F_2) + \ldots + \Pr(\mathrm{X}_N)\Pr(F_N). \quad (7)$$

If all the conditional events of F (F_i, i = 1, …, N) without exception are OMEGA events (i.e., $\Pr(F_i) = 1$) or PHI events (i.e., $\Pr(F_i) = 0$), Eq. (5) is an equivalent Sum of Disjoint Products (SDP) [2,4] of the original fault tree. This SDP reduces the probability evaluation to a simple summation.

The exact analysis of complex fault trees is a very difficult task. Many methods have been defined to reduce computation time and working memory usage. A complex fault tree is recursively decomposed into a set of mutually exclusive simpler fault trees until their dimensions are compatible with the available working memory size. Then, the results of the analysis of all generated simpler trees are composed to obtain the results for the original un-decomposed fault tree.

3. Independent Logic Trees

Fault tree analysis is a top-down approach to failure analysis, starting with a potential undesirable event called a TOP event, and then determining all the ways it can happened. The causes of the TOP event are connected through logic gates. In this paper we only consider AND-, OR-, k/n-, NAND-, and NOR-gates.

Let E_i denote that input event i occurs, and P_i = Pr(E_i). The input events are basic events or gate events. When the input events of a gate are mutually independent, we can get the exact probability of the gate event. This gate is called "independent gate" in this paper.

When we have a single gate with n mutually independent input events, we can get the exact probability of the gate using Table 1.

Table 1. Probability calculation for independent gates

gate	symbol	probability of gate
AND	*	$P_1 P_2 \dots P_n$
OR	+	$1 - (1-P_1)(1-P_2) \dots (1-P_n)$
k out of n	k	See Table 2
NAND	&	$1 - P_1 P_2 \dots P_n$
NOR	%	$(1-P_1)(1-P_2) \dots (1-P_n)$

Table 2. Probability of example k-out-of-n gates

k	n	probability of gate
2	3	$P_1P_2 + P_1(1-P_2)P_3 + (1-P_1)P_2P_3$
2	4	$P_1P_2 + P_1(1-P_2)P_3 + P_1(1-P_2)(1-P_3)P_4 + (1-P_1)P_2P_3 + (1-P_1)P_2(1-P_3)P_4 + (1-P_1)(1-P_2)P_3P_4$
3	4	$P_1P_2P_3 + P_1P_2(1-P_3)P_4 + P_1(1-P_2)P_3P_4 + (1-P_1)P_2P_3P_4$

When all the gates of a fault tree are independent logic gates, this tree is called an "Independent Logic Tree (ILT)" in this paper. We can get the exact top event probability of an ILT by simple calculations in bottom-up order.

The bottom-up algorithm is the simplest algorithm that computes the probability of failure of the top event. It starts at the bottom of the tree and compute the probability of failure of each gate based on its logic and the probabilities of its inputs. It moves upward and solves the tree at each level until the top event is reached. If a basic event appears multiple time in different places in the fault tree, the estimated probability of failure by the bottom-up algorithm is incorrect.

If a fault tree has no multiple occurring basic event (MOE), the fault tree must be an ILT. Then we can get the exact top event probability of the tree by the bottom-up algorithm.

4. Decomposition to ILTs

A large fault tree includes thousands of gates, basic events, and MOEs. The probability of the top event in a simple fault tree that does not include MOE can be computed using the bottom-up method.

In order to decompose a complex fault tree into simple ILTs, the DILT software is developed in this study.

The process of decomposing a tree to the positive and negative Shannon cofactors can be broken down into the following stages:

1. selecting a pivot basic event x_i
2. decomposing the tree with respect to x_i
3. simplifying two decomposed trees
4. ascertaining whether the decomposed tree needs further decomposing

For each decomposition of an input tree, DILT selects a pivot basic event from the MOEs of the tree. DILT decomposes the tree with respect to the selected basic event. DILT also simplifies the two decomposed trees by merging and subsuming. Then the decomposed trees are new input trees for next decompositions. If a decomposed tree is ILT, PHI event or OMEGA event, the decomposed tree is no longer a next input tree.

At the end of decomposition, there are no input tree to be decomposed. Then the exact probability of the top event of an original fault tree can be calculated by Eq. (7).

5. Application to Seismic IE Calculation

In order to assess applicability of the proposed method, an example SPSA result is selected:

- Seismic pre-ET: Fig. 1
- # of headings in ET: 7 (Fig. 1)
- Seismic induced IEs: 7 (Fig. 1)
- Seismic induced events: 26 (Table 3)
- # of random basic events in pre-ET: 2
  - SLLOCA (mean = 0.00033, EF = 4.9)
  - OPHR (mean = 0.01, EF = 10)

Table 3. SSC Fragility Results

event	A_m	β_R	β_U	event	A_m	β_R	β_U
SLOOP	0.3	0.3	0.45	ECWPF	2.78	0.35	0.34
SRTRC	0.72	0.3	0.32	ESFAF	4.41	0.38	0.85
SRTSF	1.08	0.35	0.37	SILSF	2.06	0.35	0.38
SCCSF	1.24	0.35	0.36	SLCRC	1.61	0.35	0.49
SBCRC	1.08	0.3	0.33	SLCSF	2.42	0.35	0.5
SBCSF	1.62	0.35	0.37	SBRRC	0.8	0.31	0.33
SDGCC	1.35	0.35	0.37	SBRSF	1.2	0.35	0.37
SCSTC	0.92	0.16	0.33	ECWCF	0.7	0.37	0.36
SSWRC	1.36	0.3	0.35	SECT	0.84	0.3	0.36
SSWSF	1.67	0.35	0.4	ESWXF	1.23	0.31	0.39
CCWAF	1.66	0.37	0.38	SITSF	0.98	0.21	0.46
SACBC	1.84	0.37	0.38	SSWIT	2.33	0.41	0.45
CCWPF	2.65	0.42	0.53	SICPB	1.5	0.3	0.3

The event tree of Fig. 1 is complex and non-coherent because the headings are not mutually independent. The end-states of the event tree are seismic induced IEs. The Boolean logics of the IEs (i.e., SQ 2 ~ SQ8) are written in Table 4. For example, SQ2 is a complex and non-coherent fault tree problem.

Table 5 is a set of decomposed ILTs equivalent to SQ2. Using Table 5 which is obtained by the DILT software, the exact probability of SQ2 is calculated by simple summation.

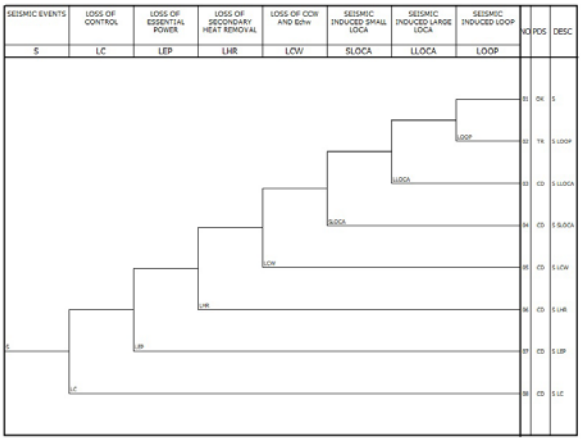

Fig. 1. Example of seismic pre-ET

Table 4. Boolean equation for ET (Fig. 1)

SQ8 * GLC
SQ7 * GLEP -GLC
SQ6 * SCSTC -GLEP -GLC
SQ5 * GLCW -SCSTC -GLEP -GLC
SQ4 * SICPB -GLCW -SCSTC -GLEP -GLC
SQ3 * GLL -SICPB -GLCW -SCSTC -GLEP -GLC
SQ2 * SLOOP -GLL -SICPB -GLCW -SCSTC -GLEP -GLC
SQ1 * -SLOOP -GLL -SICPB -GLCW -SCSTC -GLEP -GLC
GLL * SLLOCA SITSF
GLCW + GLOCCW GLOECW
GLOCCW + CCWPF CCWAF
GLOECW + GAHU ECWPF ECWCF SECT ESWXF
GAHU * ESFAF OP-HR
GLEP + SCCSF SBRSF G120VAC G480VAC G416KVAC
G120VAC * GCHAGER GXFMX
G480VAC + G480VRELAY SLCSF
G416KVAC + G416KVRELAY SSWSF
GCHAGER + GCHGRELAY SBCSF
GXFMX + GXFMRRELY SRTSF
G480VRELAY * SLCRC OP-HR
G416KVRELAY * SSWRC OP-HR
GCHGRELAY * SBCRC OP-HR
GXFMRRELY * SRTRC OP-HR
GLC + SACBC SILSF SSWIT

Table 5. Decomposed trees equivalent to SQ2
(Bottom-up ordered tree logics)

G13 + SRTRC SRTSF
G12 + SBCRC SBCSF
G09 * G12 G13
G03 * SITSF SLLOCA
G02 % SCCSF SCSTC SSWRC SSWSF CCWAF SACBC CCWPF ECWPF ESFAF SILSF SLCRC SLCSF SBRSF ECWCF SECT ESWXF SSWIT SICPB G03 G09
SQ2-1 * SLOOP G02 OPHR

G09 * SRTSF SBCSF
G03 * SITSF SLLOCA
G02 % SCCSF SCSTC SSWSF CCWAF SACBC CCWPF ECWPF SILSF SLCSF SBRSF ECWCF SECT ESWXF SSWIT SICPB G03 G09
SQ2-2 * SLOOP G02 -OPHR

Table 6 is a set of Boolean equations equivalent to SQ2 which is provided in Example PSA report. It is an input for the PRASSE code.

Table 7 is a comparison of SQ2 quantification between the 2 disjoint ILTs for DILT and the 14 disjoint equations for PRASSE, respectively. It shows that two types of Boolean equations are equivalent to each other.

Table 6. PRASSE input for SQ2 (LOOP)

1. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SRTRC
2. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF SBCSF -SRTSF -OP-HR
3. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SBCRC
4. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF SRTSF -OP-HR
5. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF SBCRC -SRTRC
6. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SBCRC
7. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF SITSF -SLLOCA -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF -OP-HR
8. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SRTRC
9. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF SBCSF -SRTSF -OP-HR
10 -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SBCRC
11. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF SRTSF -OP-HR
12. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF SBCRC -SRTRC
13. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF OP-HR -SLCRC -SSWRC -ESFAF -SBCRC
14. -SCSTC -SICPB SLOOP -SACBC -SILSF -SSWIT -SCCSF -SBRSF -SITSF -SLCSF -SSWSF -CCWPF -CCWAF -ECWPF -ECWCF -SECT -ESWXF -SBCSF -SRTSF -OP-HR

Table 7. SQ2 probabilities using Table 5 and 6

Pr(E_i)	DILT	PRASSE
0.1	1.9593734E-02	1.9593734E-02
0.3	9.2428376E-04	9.2428376E-04
0.5	4.6044588E-06	4.6044588E-06
0.7	8.0039493E-10	8.0039497E-10
0.9	3.2520647E-18	3.2520761E-18

5. Conclusions

The exact analysis of complex and non-coherent Boolean equations (e.g., event tree) is a very difficult task. This study developed a method to exactly quantify complex Boolean equations based on the Shannon

decomposition technique and the characteristics of Independent Logic Tree (ILT). This method was implemented in the DILT software. DILT can treat the 5 types of logic gates: AND-, OR-, k/n-, NAND-, and NOR-gates. Non-coherent fault trees can be expressed by NAND-, and NOR-gates. The DILT software converses an original tree into the equivalent set of decomposed ILTs and disjoint products. The output of DILT is easily understandable and simpler than disjoint equations of the PRASSE code.

REFERENCES

[1] R. Bryant, Graph Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers, C-35(8), 677-691, 1986.
[2] Rauzy A, New algorithms for fault trees analysis, Reliability Engineering and System Safety, 40, 203–211, 1993.
[3] Jung S, Han SH, Yang JE. Fast BDD truncation method for efficient top-event probability calculation, Nuclear Engineering Technology, 40, 571–80, 2008.
[4] J. Choi, N. Cho, A Practical Method for Accurate Quantification of Large Fault Trees, Reliability Engineering and System Safety, 92, 971-982, 2007.