TREATMENT OF LOGICAL LOOPS IN PSA FOR RISK BASED DECISION MAKING

Joon-Eon Yang, Sang-Hoon Han and Young-Ho Jin
Korea Atomic Energy Research Institute, Korea

ABSTRACT

In most PSAs implementing the fault/event tree approach, there are several limitations to get the proper minimal cut sets. One such limitation is the logical loop, which occurs due to mutual dependencies between support systems. Until now, the logical loop has been removed by ignoring some relations between the systems. However, this conventional method might generate some nonsense cut sets or miss some important cut sets since the real relations between the systems are ignored. This kind of error can lead to misinterpretation of PSA results and, therefore, improper decisions. An analytical method to solve the logical loop exactly has been developed by the Korea Atomic Energy Research Institute (KAERI) and implemented in a cut set generation code called the KAERI Reliability Analysis Package (KIRAP). In this method, the Boolean logic of the fault tree is regarded as a kind of network in which the events/gates are regarded as the nodes. The paths of this network are examined by a top-down approach, and the path, which causes the logical loop, is identified and deleted based on the Boolean criteria. The logic of the fault tree is therefore changed into new logic without logical loops while maintaining proper relations between systems.

INTRODUCTION

Recently, Probabilistic Safety Assessment (PSA) has been widely used as a tool to support the decision making in various fields of nuclear industry, such as operation and maintenance. For instance, the in-service test interval can be determined based on the PSA results. In order to make proper decisions based on the results of PSA, we should have the proper results. The event and fault tree method is used in most PSA and, in such cases, we get the minimal cut sets as results. A minimal cut set means a combination of events that results in the failure of a system or a sequence. However, there are several limitations to get the proper minimal cut sets, such as the truncation limit used in generating the cut sets, etc. One such limitation is the logical loop, which occurs due to mutual dependencies among the support systems. For instance, service water system supports the electric power supply system, and vice versa. Therefore, when we model these systems by fault trees, logical loops are generated in the fault tree model. Since most cut set generation computer codes cannot handle the fault tree logic with the logical loop, until now the logical loop has been removed manually by ignoring some relations between the systems. For this, the analyst has to develop other fault trees of the systems without logical loops and use these fault trees instead of the original ones to remove the logical loop (Coles & Powers, 1989).

However, this conventional method might generate some nonsense cut sets or miss some important cut sets, since the real relations between the systems are ignored. Such errors can cause a problem, not only when we calculate the reliability of a system, but also when we calculate the frequencies of accident sequences. Therefore, this kind of error can lead to a misinterpretation of the PSA results. For instance, if a minimal cut set representing the relation between system A and B is not generated, a plant's staff could make a decision to perform a maintenance of system A, thinking that system A does not affect the reliability of system B. However, it is not a proper decision. Another approach to solve the logical loop is the iteration method, that has been developed by the Korea Atomic Energy Research Institute (KAERI). In this method,

first we assumed that there are no dependencies between support systems, and we get the Boolean equations for each support system. During the next step, we updated the Boolean equations of the support systems using the Boolean equations of the previous step. Such iterations continue until the Boolean equations of each support system converge to some final Boolean equations. In order to use this method, it is necessary to repeat the quantification processes and to review the generated cut set at each iteration step. It is difficult and time-consuming work.

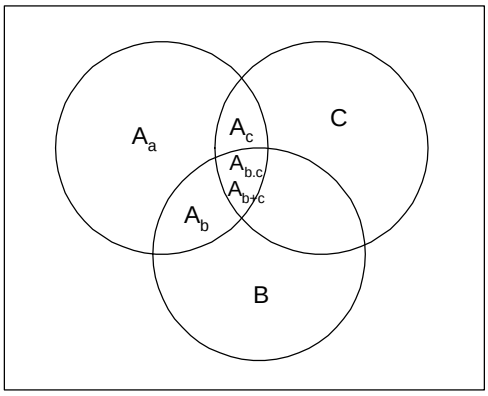
Therefore, an analytic method to solve the logical loop exactly and efficiently has been developed by KAERI and implemented in a cut set generation code called the KAERI Reliability Analysis Package (KIRAP) (Yang & Han, 1997). This method enables us to modify the fault tree logic automatically and exactly in one quantification step based on the Boolean logic of the fault trees. In this method, the Boolean logic of the fault tree is regarded as a kind of network in which the events/gates are regarded as the nodes. The paths of this network are examined by a top-down approach, and the path that causes the logical loop is identified and deleted based on the Boolean criteria. The logic of the fault tree is therefore changed into new logic without logical loops while maintaining proper relations between systems. We compare the minimal cut sets obtained by the conventional and new methods in order to show the advantages of the new method over the conventional one.

ANALYTICAL METHOD TO BREAK THE LOGICAL LOOP

Boolean Expression of Fault Tree Logic with the Logical Loops

For instance, let us assume that there are three support systems: A, B and C (Figure 1).

Figure 1 The Relations among Three Support Systems: A, B and C



The failure of support system A, then, can be represented by the following Boolean equation:

$$A = A_a + A_b.B + A_c.C, \quad \text{..... (1)}$$

where A = a set representing the failure of system A,

A_a = a subset representing the failures of system A's part that cause the failure of system A without the simultaneous failure of system B or C,

A_b = a subset representing the failures of system A's part that cause the failure of system A when the simultaneous failure of system B occurs,

A_c = a subset representing the failures of system A's part that cause the failure of system A when the simultaneous failure of system C occurs,

+ = OR operation, and

. = AND operation.

However, in some cases, there can be some common parts between subset A_b and A_c. So, to ensure that there are no common parts between A_b, and A_c, we restructure the above equation into the following:

$$A = Aa+Ab.B+Ac.C+Ab+c.(B+C)+Ab.c.B.C, \quad \text{..... (2)}$$

where A_{b+c} = a subset representing the failures of system A's part that cause the failure of system A when the failure of system B or C occurs,

$A_{b.c}$ = a subset representing the failures of system A's part that cause the failure of system A when the failures of system B and C occur at the same time.

Hence, the following constraint can be applied to A_a, A_b, A_c, A_{b+c} and $A_{b.c}$:

$$Ai.Aj = \Phi \text{ (the empty set) } (i,j = a, b, c, b+c, b.c; i \neq j). \quad \text{..... (3)}$$

This constraint means that there are no common parts among the subset A_b, A_c, A_{b+c} and $A_{b.c}$.

In the same way, the failures of system B and C can be represented by the following Boolean equations, respectively:

$$B = Bb+Ba.A+Bc.C+Ba+c.(A+C)+Ba.c.A.C, \quad \text{..... (4)}$$

$$C = Cc+Ca.A+Cb.B+Ca+b.(A+B)+Ca.b.A.B. \quad \text{..... (5)}$$

Analytical Method to Break the Logical Loops Automatically

Let us call the subsets such as A_a, B_b and C_c the basic event sets, and the subsets, such as A_b, A_c, A_{b+c}, $A_{b.c}$, etc., the conditional event sets. The basic event set of each system represents the failures of each system's main parts causing the failure of each system without simultaneous failure of related support systems, e.g., the motor failure of a pump. The conditional event set represents the failure of a system's part that can cause the failure of the entire system when the related support system fails simultaneously, e.g., the simultaneous failures of the instrument air system and the local backup air tank for a system. The element of a basic event set or the elements of a conditional event set when combined with those of other support systems' basic event sets can cause the failure of that system.

If we expand equations (2), (4) and (5) for system A, we will get many terms with the form shown below:

$$Ai.[B \text{ or } C].A. \quad \text{..... (6)}$$

where Ai = an element of the conditional event set of system A, e.g., Ab or Ac or Ab+c or Ab.c,

[B or C] = the conditional event set for system B or C that appears on the right hand side of equations (4) and (5).

Equation (6) represents the relation where a failure of system A causes the failure of system B or C, which in turn, causes the failure of system A again; that is, the logical loop. For the case where A_i is a basic event, e.g., A_a, this term is reduced by a Boolean reduction rule X+X.Y=X] since A_a already exists in equation (2).

Let us expand the above equation for a case where A_i is A_b:

$$A_b.[B \text{ or } C].A = A_b.(A_a+A_b.B+A_c.C+A_{b+c}.(B+C)+A_{b.c}.B.C).[B \text{ or } C]$$

$$= A_b.\underline{A_a}+A_b.B+\underline{A_b.A_c}.C+\underline{A_b.A_{b+c}}.(B+C)+\underline{A_b.A_{b.c}}.B.C).[B \text{ or } C]$$

(the underlined terms become Φ by equation (3))

$$= A_b.B.[B \text{ or } C]$$

$$\therefore \{Ab.[B \text{ or } C]\}.A = \{Ab.[B \text{ or } C]\}.B \quad \text{..... (7)}$$

The above Boolean equation is valid only for the following three cases, since we assume that A and B are different systems:

$$\text{i)} \quad Ab = \Phi, \quad \text{..... (8)}$$

$$\text{ii)} \quad [B \text{ or } C] = \Phi \text{ and} \quad \text{..... (9)}$$

$$\text{iii)} \quad Ab.[B \text{ or } C] = \Phi \text{ or } Ai.[B \text{ or } C].A = \Phi. \quad \text{..... (10)}$$

For the first and second cases, these terms would not be generated during the expansion. Therefore, if we get these kind of terms while we expand the above Boolean equations, only the relation of the third case is correct. So, in such a case, we do not need to expand these terms, but need only to delete them.

The Boolean equations for m support systems can be expressed as the following:

$$ni = ni^* + \Sigma j \text{Cond } i(j) f\ i\ j(n1,n2,...,ni-1,ni+1,...,nm) \quad \text{..... (11)}$$

$$nj = nj^* + \Sigma k \text{Cond } j(k) fjk(n1,n2,...,nj-1,nj+1,...,nm) \quad \text{..... (12)}$$

where ni = a set representing the failure of the i-th support system,

ni*= a basic event set of the i-th support system,

Cond i(j) = the j-th conditional event set of the i-th support system and

f i j(n1,n2,...,ni-1,ni+1,...,nm) = the combination of n1,n2,...,ni-1,ni+1,...,nm corresponding to the j-th conditional event set.

Since the intersection of each conditional event is an empty set, as shown in the above example, when we expand the above equations for n_i, we will get terms like the following:

$$\text{Cond } i(k).[\text{terms of } nj \neq i].nI \quad \text{..... (13)}$$

$$= \text{Cond}_i(k).[\text{terms of } n_{j \neq i}].\{ n_i^* + \Sigma_j \text{Cond}_i(j) f_{ij} (n_1,n_2,...,n_{i-1},n_{i+1},...,n_m)\}$$

$$= \text{Cond } i(k).[\text{terms of } nj \neq i].nk \quad \text{..... (14)}$$

The term $\text{Cond}_i(k).[\text{terms of } n_{j \neq i}] .n_i$ should be an empty set for the same reason as above.

To implement the above method, we expand the logic of the original fault tree using the top-down approach. During the expansion, we can use equation (14) as the criteria to decide which terms cause the logical loops and should be deleted. By expanding and deleting the logic of the fault trees, we will get the new logic of the fault trees without the logical loops while maintaining proper relations among the support systems. These new fault trees are used to generate the minimal cut sets for the support systems.

APPLICATION TO AN EXAMPLE PROBLEM

The developed method is tested for an example case. We quantify the sample core damage frequency (CDF) based on two typical event trees of CANDU PSA: a Small Loss of Coolant Accident (LOCA) and a General Transient. The front-line systems are shown in Table 1 with their dependencies to support systems. The dependencies between the support systems are shown in Figure 2. Simplified fault trees for the front-line and support systems are developed to quantify the sample CDF. The Boolean equations of some fault trees are shown below:

FW = FW-HW + (RCW * FW-RCW-BKUP + IA * FW-IA-BACKUP + NEPS).

RSW = RSW-HW + (IA + NEPS).

RCW = RCW-HW + (RSW + IA + NEPS).

IA = IA-HW + (RCW + NEPS).

NEPS = NEPS-HW + (RSW * NESP-RSW-BKP).

where FW = Failure of Feedwater System,

NEPS = Failure of Normal Electric Power Supply System,

EPS = Failure of Emergency Power Supply System,

RCW = Failure of Recirculated Cooling Water System,

RSW = Failure of Raw Service Water System,

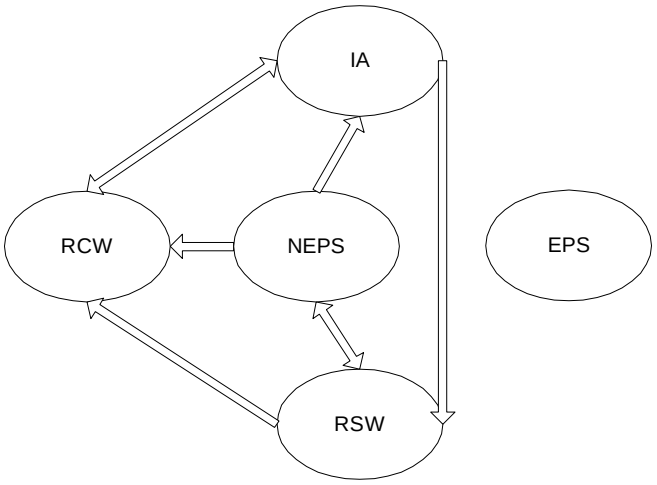
IA = Failure of Instrument Air System.

In the above Boolean equations, the post-fix -HW represents the failure of hardware, and -BKP represents the failure of the backup system. For instance, in the Boolean equation for NEPS, NEPS-HW represents the hardware failure of the normal electric power supply system, RSW represents the failure of the raw service water system and NESP-RSW-BKP represents the failure of the RSW backup system for the normal electric power supply system. An example of a fault tree for an electric power supply system is shown in Figure 3.

Table 1 Dependency Table of Front Line Systems

System	NEPS	EPS	RCW	RSW	IA
Reactor Shutdown	O	X	X	X	X
Steam Generator Pressure Relief & Control	O	X	X	X	O/BKP
Crash Cooldown	O	X	X	X	O/BKP
Feedwater System	O	X	O/BKP	X	O/BKP
Emergency Water Supply System	O	O	O	X	O/BKP
Condensate System	X	X	X	X	X
Shutdown Cooling System	O	X	O	X	X
Emergency Core Cooling System	O	O	O/EWS	X	X
Moderator Heat Sink	O	X	O	X	X

Figure 2 Dependencies and Logical Loops among Support Systems



Loop 1: IA--RCW--IA
Loop 2: NEPS--RSW--NEPS
Loop 3: NEPS--IA--RSW--NEPS
Loop 4: NEPS--RCW--IA--RSW--NEPS
Loop 5: RSW--RCW--IA--RSW

As shown in Figure 2, there are five logical loops between the support systems. These logical loops are to be broken to quantify the sample CDF. We have solved these logical loops using three methods and compared the results of each method. The three methods are (1) the developed method, (2) the iteration method and (3) the modified fault tree method. We briefly explain the latter two methods below.

- **Iteration Method:**

This method has also been developed by KAERI. In this method, we first assume that there are no dependencies between support systems, and we get the Boolean equation for each support system. During the next step, we update the Boolean equation of the support systems using the Boolean equation of the previous step. Such iterations continue until the Boolean equations of each support system converge to some final Boolean equations.

- **Modified Fault Tree Method:**

In this method, we identify the logical loops between support systems and also identify the breaking points that have a minimal effect on the quantified results. Based on the breaking points between support systems, we develop new fault trees without logical loops and these fault trees are used to quantify the final results instead of the original ones. The modified fault tree can be used in two ways: one is to link these fault trees directly to the front line systems and the other is to link these fault trees to other support systems. For this example case, we adopt the later approach.

An example of a fault tree is shown in Figure 3. This fault tree shows a logical loop between NEPS and RSW. In the developed method, the NEPS gate marked as a gray box in Figure 3 is identified as a cause of logical loop and deleted automatically. On the other hand, in the modified fault tree method, a new fault tree without logical loops is developed for NEPS and used instead of the original NEPS fault tree in order to remove the logical loop as shown in Figure 4.

Figure 3 Fault Tree for NEPS with Logical Loop

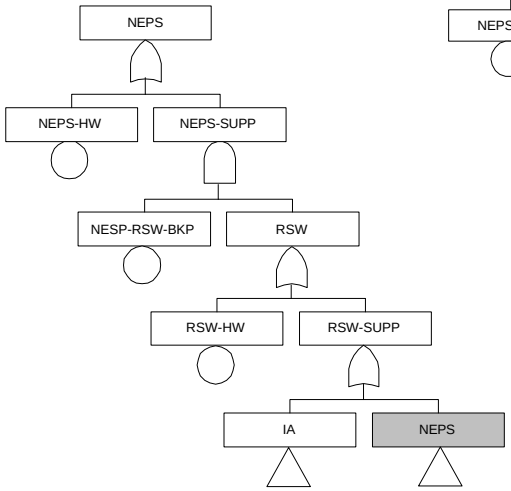
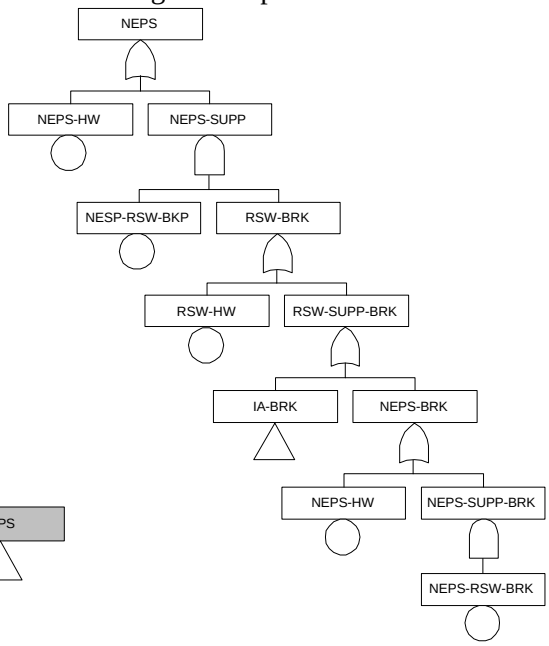


Figure 4 Modified Fault Tree for NEPS without Logical Loop

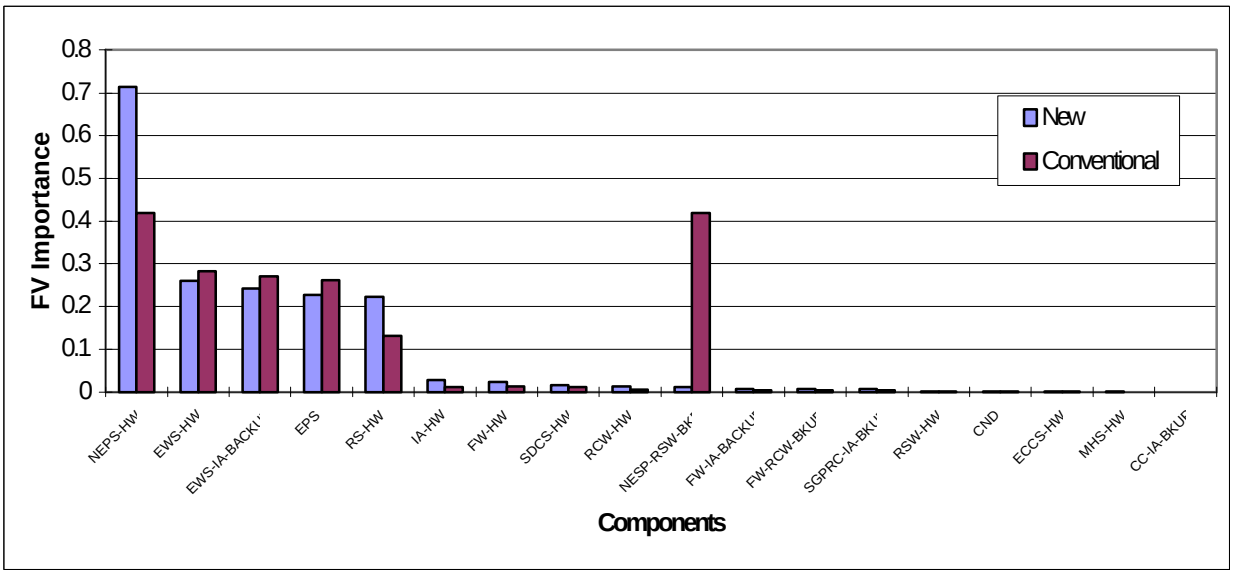


The differences of quantification results using the above three methods are shown in Table 2. The developed and iteration methods give the exact same results. The modified fault tree approach, however, generates some nonsense cut sets, i.e., the wrong CDF value, and these cut sets also affect the importance of basic events as well (Figure 5).

TABLE 2 QUANTIFICATION RESULTS

Developed and Iteration Methods			
NESP-RSW-BKP	IA-HW	EPS	IE-GT
NESP-RSW-BKP	IA-HW	EWS-IA-BACKUP	IE-GT
NESP-RSW-BKP	IA-HW	EWS-HW	IE-GT
Conventional Method			
NESP-RSW-BKP	EPS	IE-GT	
NESP-RSW-BKP	EWS-IA-BACKUP	IE-GT	
NESP-RSW-BKP	EWS-HW	IE-GT	

Figure 5 Fussel-Vesely Importance of Basic Events



CONCLUSIONS & DISCUSSION

An analytical method has been developed and applied to an example problem to solve logical loops between support systems. The result shows that the conventional method might cause some nonsense cut sets. Such nonsense cut sets are generated due to the arbitrary breaks of fault trees for support systems. We cannot say that the conventional method always gives improper results. It depends upon the structures and relations between the fault trees of support systems. In the conventional method, it is very difficult and time-consuming to develop new fault trees to break the logical loops and to review the generated cut sets, whether or not nonsense cut sets exist. Such nonsense cut sets might cause inaccuracy and misinterpretation of the final PSA results. For instance, as shown in Figures 5, the importance of the hardware failure of an electric power system is underestimated and the RSW backup system for an electric power system is overestimated. Of course, these values are based on the assumed hardware failure rate, therefore this importance analysis is not an exact one. However, this example shows that there is a possibility that the conventional method could lead to misinterpretation of PSA results.

Even though the iteration method gives exact results, the method requires iteration of the quantification process. It is necessary to review the minimal cut sets for each iteration step to check whether or not the minimal cut sets for each support system have converged. This method therefore also requires a lot of manpower.

Comparing the above two methods, the developed method gives an exact result in one quantification process and guarantees the complete minimal cut sets. By implementing the developed algorithm to solve the logical loop, therefore, we don't need to worry about inaccuracy and/or incompleteness of PSA results caused by the logical loops, and we can avoid errors based on improper PSA results. For instance, when we select the items for the in-service test based on PSA results, the nonsense cut sets caused by logical loops might have a big effect, since such cut sets are related to support systems. Or, when we use a risk monitor, there can be a configuration change in a support system. This might change the structure of logical loops, then conventional method might cause a problem. The developed method, however, can handle such case as well.

In addition to logical loops, there are a number of other problems, which cause the inaccuracy, and/or incompleteness of PSA results. The developed algorithm is a means of eliminating one such problem to derive proper PSA results and/or proper decisions based on such results.

REFERENCES

Coles, G.A. and Powers,T.B. “Breaking the Logic loop to Complete the Probabilistic Risk Assessment”: *Proceedings of PSA ’89, International Topical Meeting Probability, Reliability, and Safety Assessment,* Pittsburgh, 1989*, p.*1155, American Nuclear Society, La Grange Park, IL, 1989

Yang, J.E. and Han, S.H. et al. "Analytic Method to Break Logical Loops Automatically in PSA". *Reliability Engineering and System Reliability.* Vol. **56**, *No. 2, p.*101 , 1997.

KEY WORDS

Logical Loops, Support Systems, Dependency, Boolean Criteria, Risk Based Decision Making.