Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 10-11, 2007

Characteristics of a Circular Logic and Its Treatment

Ho-Gon Lim[a], Sang-Hoon Han[a], Joon Eon Yang[a]
[a] *Integrated Safety Assessment Division, Korea Atomic Energy Research Institute, (150-1 Dukjing-Dong), 1045 Daedeokdaero, Yuseong, Daejeon, Korea., hglim@kaeri.re.kr*

1. Introduction

A circular logic or a logical loop is defined as the infinite circulation of supporting relations due to their mutual dependencies among the systems in the Fault Tree Analysis (FTA). While many methods to break the circular logic have been developed and used in the fault tree quantification codes, the general solution for a circular logic and its breaking methods are not generally known as yet. This paper presents an analytic solution for circular logics in which the systems are linearly interrelated with each other. Then, a general treatment of circular logics is discussed. To formulate the analytic solution, the relations among systems in the fault tree structure are described by the Boolean equations. The solution is, then, obtained from the successive substitutions of the Boolean equations, which is equivalent to the attaching processes of interrelated system's fault tree to a given fault tree. The solution for three interrelated systems and their independent fault tree structures are given as an example.

2. Analytic Solution

2.1 Deduction of the analytic solution

If the "n" numbers of systems constitute circular logics with linear dependencies, the following equations can be written for N interrelated systems in terms of the Boolean expression.

$$\begin{aligned} S_1 &= c_1 + a_{12}.S_2 + a_{13}.S_3 + \Lambda + a_{1n}.S_n \\ S_2 &= c_2 + a_{21}.S_1 + a_{23}.S_3 + \Lambda + a_{2n}.S_n \\ &\text{M} \quad \text{M} \quad \text{M} \quad \text{M} \quad \Lambda \quad \text{M} \\ S_n &= c_n + a_{n1}.S_1 + a_{n2}.S_2 + \Lambda + a_{nn-1}.S_{n-1}. \end{aligned} \tag{1}$$

where S_i, c_i, and a_{ij} are a total failure events of system i, basic events representing an independent failure of system i, and common events causing the failure of system i, respectively. + and · operations in Eq. (1) mean "OR" and "AND" operation of Boolean algebra respectively.

Eq. (1) can be solved by a component wise matrix form as shown in Eq. (2)[1]

$$\begin{aligned} S_i = c_i + \sum_{\substack{j_1=1 \\ j_1 \neq i}}^{n} a_{ij_1} \Bigg(& c_{j_1} + \sum_{j_2=i} a_{j_1 j_2} \delta_{j_2} + \\ & \sum_{\substack{j_2=1 \\ j_2 \neq i, j_1}}^{n} a_{j_1 j_2} \Big(c_{j_2} + \sum_{j_3=i, j_1} a_{j_2 j_3} \delta_{j_3} + \Lambda \\ & + \sum_{\substack{j_{n-1}=1 \\ j_{n-1} \neq i, j_1, \Lambda\, j_{n-2}}}^{n} a_{j_{n-2} j_{n-1}} \Big(c_{j_{n-1}} + \sum_{j_n=i, j_1, \Lambda\, j_{n-2}} a_{j_{n-1} j_n} \delta_{j_n} \Big) \Lambda \Bigg) \end{aligned} \tag{2}$$

Where the term δ_j is determined by the fault tree analyst depending whether the self recursion terms have a meaningful event set or not. The assignment of a value for δ_j is given as follows:

$$\begin{cases} \delta_i = \textit{false if the self recursion term has no event set} \\ \delta_i = \textit{ture if the self recursion term has a event set} \end{cases}$$

2.2 Application for three interrelated systems

We show a simple example for fault tree structure of three interrelated systems.

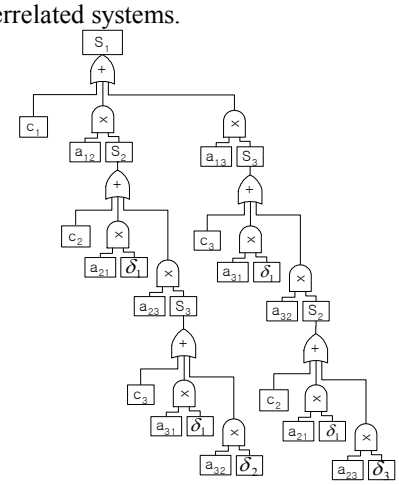

Figure 1 solution for three interrelated systems

The analytic solution for the three interrelated systems is provided from Eq. (2) by letting $n=3$.

$$\begin{aligned} S_i = c_i + \sum_{\substack{j_1=1 \\ j_1 \neq i}}^{3} a_{ij_1} \Bigg\{ & c_{j_1} + + \sum_{j_2=i} a_{j_1 j_2} \delta_{j_2} + \\ & \sum_{\substack{j_2=1 \\ j_2 \neq i, j_1}}^{3} a_{j_1 j_2} \Big(c_{j_2} + \sum_{j_3=i, j_1} a_{j_2 j_3} \delta_{j_3} \Big) \Bigg\} \end{aligned} \tag{3}$$

For system *1*, the following Boolean equation will be obtained by letting $i=1$ and expanding summation convention for each index as follows.

$$S_1 = c_1 + a_{12}\left(c_2 + a_{21}\delta_1 + a_{23}\left(c_3 + a_{31}\delta_1 + a_{32}\delta_2\right)\right) + a_{13}\left(c_3 + a_{31}\delta_1 + a_{32}\left(c_2 + a_{21}\delta_1 + a_{23}\delta_3\right)\right) \quad (4)$$

From Eq. (4), the independent fault tree of system *1* can be constructed as shown in Fig. 2. As shown in Fig. 2, for three interrelated systems, two attaching processes are needed to eliminate the dependencies. At each attaching step, the self recursion events should be handled properly according to their physical meaning on the system failure

3 Treatment of Circular Logic

When a system is modeled by mathematical way, initial conditions for the system are necessary to know the evolution of the system status. However, in a fault tree model of the systems, initial condition is not explicitly given for the development of sub-system or component. These approaches do not mainly invoke critical problem when a system is independent on any other system. However, when a system is coupled with other systems to make circular logics, the initial condition may be important factor to determine the system's failure event. If a fault tree for a system is developed, the fault tree can have a two initial condition, that is, a standby state or running state. We have divided these two cases and the treatment methods of circular logic are explained separately.

2.1 When a system is in standby

Figure 2 shows a circular logic extracted from the Ulchin 3&4 PSA model, so called PRIME-U34i.[2] This circular logic is for AAC DG (diesel generator) failure event.

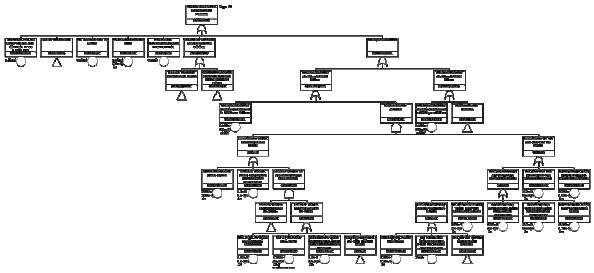

Figure 2 Circular Logic for AAC DG Standby failure

The circular logic of figure 2 is composed of the following sequences. PCS (plant control system) do not give signal to AAC DG. PCS is failed by EPS (electric power supply) for PCS. The electricity to PCS can be supplied by two ways. One is battery and the other is 480V MCC (motor control center). Then, 480V MCC is failed by AAC DG. In a real situation, AAC DG cannot receive actuation signal if battery does not give electricity to PCS. If the terminal gate of AAC DG failure is considered as an initial condition, this circular logic can be solved easily as in a real situation. Since the AAC DG is in standby, that is, in a false state, the AAC DG failure event at the terminal of the fault tree give false event condition. In a Boolean algebra, this means that the AAC DG failure event is treated as universal event. By this treatment, the circular logic can be broken and the fault tree can describe real event situation.

2.2 When a system is running

Figure 3 shows a circular logic extracted from PRIME-U34i.

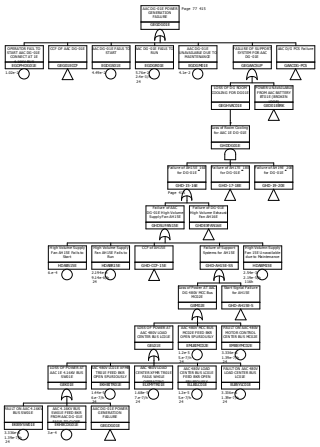

Figure 3 Circular Logic for AAC DG running failure

The circular logic of figure 3 is composed of the following sequences. HVAC (Heating, Ventilation & Air Conditioning) is failed to give room cooling for AAC DG by EPS. EPS is failed by AAC DG failure. In a real situation, AAC DG can be started without HVAC. As in the case of previous section, if AAC DG failure event in the terminal gate is treated as null event, the fault tree can describe real situation.

3. Conclusion

This paper presents an analytic solution for circular logics in which the systems are linearly interrelated with each other. Then, a general treatment of circular logics is discussed. We expect that if an initial state of a system is incorporated into the fault tree model, an autonomous treatment of circular logic in a large fault tree can be easily done.

REFERENCES

[1] H. G. Lim, S. C. Jang, 2007, An analytic solution for a fault tree with circular logics in which the systems are linearly interrelated systems, Relia. Eng. & Sys. Safety, Volume 92, P. 804-807

[2] KAERI, 2004, Development of Risk-Informed Application Technology, KAERI/RR-2496/2004