Reliability evaluation of FTA with feedback loop

Takeshi Matsuoka1[1)]

1) Collaboration Center for Research and Development, Utsunomiya University,
350 Mine, Utsunomiya city, Tochigi, Japan, 321-8505 (mats@cc.utsunomiya-u.ac.jp)

***Abstract*— This paper presents a procedure to solve FT with feedback loop in success events expression. First, explain the basic procedure by using simple dependent fault trees. Detailed analysis steps are described. Next, the procedure is applied to more complex fault trees with 3 non-linear interrelated loops. In the analysis procedure, FT structures are converted into a physical system model, which satisfy Boolean equations in success events expression. The Boolean equations are solved with the aid of takeover phenomenon by considering the process of establishing loop structured operation. Finally, the solution is converted into failure expression and obtain a fault tree structure.**

It is shown that the representative possible solution of Fault Tree with feedback loop can be obtained by formulary method. A fault tree with feedback loop can be solvable without approximation. The proposed method is particularly useful in evaluating reliability and/or availability of any kinds of complex engineering systems.

Keywords—Fault tree analysis; FT with feedback loop; Logical loop structure; Boolean equation; Exact method.

I. INTRODUCTION

Many kinds of techniques are used for high reliable and safety systems, for example redundant components, back-up function by stand-by component, principle of diversity, and so on. One of the distinguished designs of nuclear power plant is as follows. A main functional system is supported by multiple subsystems, which provide electricity or cooling water or lubrication oil. These support systems are sometimes mutually supported and/or recursively supported by main system. This system configuration leads to a problem of solving Fault Trees with feedback loop.

Solving Fault Tree (FT) means to obtain all the possible minimal cut sets. If there are some missing minimal cut sets, there is a danger to miscalculate top event probability, because missing cut set has a possibility to have large contribution.

Obtaining minimal cut sets is easy for FTs without feedback loop. First, simply make products (cut sets) based on the FT structure, and eliminate sub cut sets. Then all the minimal cut sets can be obtained.

For FT with feedback loop, the above first step can not be done because top event recursively appear and cut sets are endlessly produced. If reappeared top event is ignored at certain point (most of the proposed methods to solve FT with loop), we can obtain some group of cut sets. But the result is not assured if there are some missing minimal cut sets or not, and it is approximate solution.

Conventional methods [1-4] propose to solve the logical loop problem by breaking the logical loops at the points where the dependencies among the support systems are relatively weak and developing new fault trees without the logical loops. But this method does not give us exact solution of Fault Trees with loops. Yang [2] built contra-example for this approach, which shows its mistake. FT can be expressed by Boolean equations, and they are mathematically solvable with arbitrary sets [5].

For solving FT, there is a method by binary decision diagram (BDD) [6] in which FT is solved without making minimal cut sets, but by directly making combinations of success and failure events for occurrence of top event. This method also cannot be applicable to the FT with feedback loop. Factor graph method [7] has been also presented for solving a system with loop. In this method, endless connection is terminated at a certain point.

This paper presents procedure to solve Fault Trees with feedback loop in success event expressions.

In Section 2, give the analysis condition and classification of components. In section 3, take up a simple example of mutually dependent fault trees and detailed analysis procedures are shown. Failure event relationship is converted into success event relationship and solved with an arbitrary set "*m*" by Boolean arithmetic calculation. The arbitrary set "*m*" is determined by considering a physical model and takeover phenomenon. Then, the solution is converted into failure event relationship and obtain a fault tree expression. In section 4, the proposed procedure is applied to more general loop structured system and confirm that this procedure is generally applicable to mutually dependent fault trees. Finally, conclusions are given in section 5.

II. ANALYSIS CONDITIONS

As fault tree structure, gates are expressed by two main logic gates – AND-gate and OR-gate. In the analysis by success event expression, change of system states with time is considered and it is assumed that all the components are placed in standby state at initial time, and they are started at designated time. If a component fails, it cannot be repaired, that is, non-repairable model is taken up. Components are classified into three types, a self-sustained type (called as "SS-type" in this paper), a generative type (G-type) and a transmitter type (T-type), which adequately explain the operational state of a loop structured system [8].

III. SIMPLE DEPENDENT FAULT TREES

First, consider a simple fault tree structure and brief procedure is explained for solving mutually dependent fault trees.

$$A = A_a + A_b \cdot B \quad (1)$$
$$B = B_b + B_a \cdot A, \quad (2)$$

where A and B are top events, A_a, A_b, B_a and B_b are non-repairable basic events.

A. Analysis in fault tree expression

Two fault trees are combined into one fault tree with only one recursion term A, but this combined fault tree has endless recursion as shown in Fig.1.

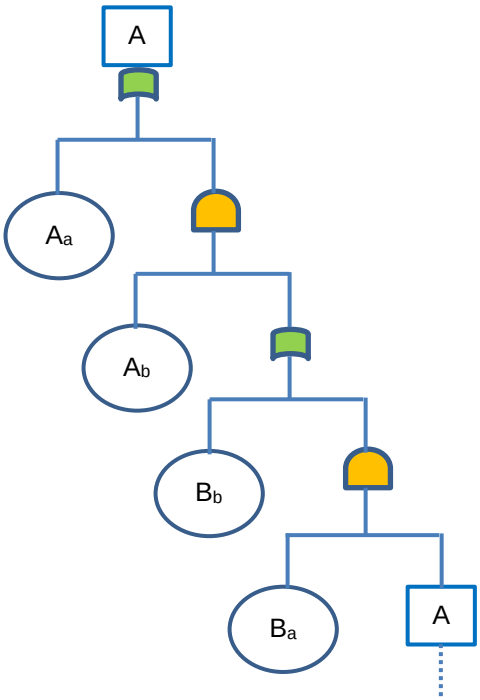

Fig.1 A fault tree with one recursion term A.

A.1. Analysis by simple cutoff method

The algorithm is as follows. A re-appeared top event is just eliminated, and solution is obtained as follows in this case.

$$A = A_a + A_b \cdot B_b \quad (3)$$

A.2. Analysis by Yang's method [2]

The essence of the algorithm is described as follows. Repeat the substitution of fault tree. During this process when we get some cycle *A-B-A*, stop expansion on this direction and to delete this cut set. The solution becomes Equation (3).

A.3. Analysis by Vaurio's recursive method [3]

Start with the assumption $A=B=False=\Phi$. Substitute them into Equations. (1) and (2), and get new values of top events A and B. Repeat this process until leading to a stable unchanging solution. The solution becomes Equation (3).

B. Analysis in Success Event Expression

Relations expressed in failure events can be converted into relations expressed in success events by using basic Boolean rules. Let denote a as a complement of failure event A. Then the Equation (1) and Equation (2) can be converted into following two expressions.

$$a = a_a \cdot b + a_a \cdot a_b \quad (4)$$
$$b = b_b \cdot a + b_b \cdot b_a, \quad (5)$$

Substitute Equation (5) into Equation (4),

$$a = a_a \cdot b_b \cdot a + a_a \cdot b_b \cdot b_a + a_a \cdot a_b \quad (6)$$

The left side term "a" is defined by using the term "a" itself.

B.1. Structural Relation of Success Events

The Boolean relations given by Equation (4) and Equation (5) are expressed by Fig. 2. Where "a_b" is a success event associated with some physical element. An arrow means a success event makes product with endpoint event. Product "$a_a \cdot a_b$" is produced. But in Fig.2, success event "a_a" is also related to "b", and "b" and "a_b" are parallel input to "a_a". Then, the resultant event, or output of some physical element, which is denoted as "a", has contributions from the "$a_a \cdot a_b$" and "$a_a \cdot b$".

We consider a physical system model which corresponds to Fig.2, as shown in Fig.3. In this figure, components are explicitly expressed, and the successful operation of component A_b. is denoted as success event "a_b".

Components A_b and B_a support components A_a and B_b, respectively, where components A_b and B_a are SS-type components denoted by yellow color and components A_a and B_b are G-type or T-type components [8]. The operation of SS-type component needs not require any support. The operation of G-type or T-type component requires support by other component.

The component A_a has two supports "a_b" and "b", then the output of component A_a (event "a") becomes Equation (4). Where, event "a_a" expresses elementally characteristics of component A_a itself, and does not express the actual operating state of component A_a, but expresses some soundness of component A_a [9].

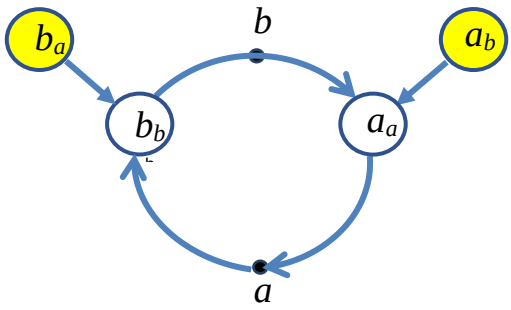

Fig.2 Relations between success events in Eqs (4) and (5).

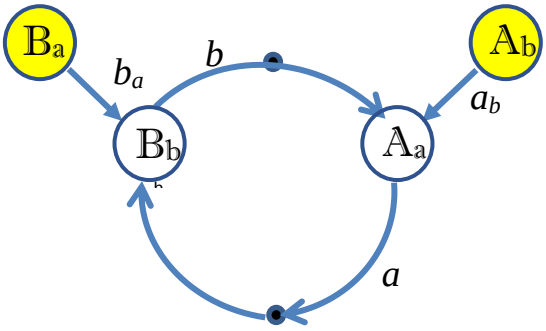

Fig.3 Physical system model corresponds to Fig. 2.

It is possible to make other three different physical models which correspond to Boolean relations (4) and (5). The endless recursive situation, which appears in the fault tree expression, is not appeared in Figs. 2 and 3. The term "$a_a \cdot b_b \cdot a$" in Eq. (6) corresponds to a loop from "*a*" to "*a*" via "b_b" and "a_a" as seen in Fig. 2. It also corresponds to a loop from "*a*" to "*a*" via "B_b" and "A_a" as seen in Fig. 3.

B.2. Solution by Boolean Equation

Output "*a*" is expressed by Equation (6) as shown in the section *B.* and it can be solved as follows [10],

$$a = m \cdot a_a \cdot b_b + a_a \cdot b_b \cdot b_a + a_a \cdot a_b \qquad (7)$$

where *m* is an arbitrary set in mathematical meaning, and it is determined depending on the starting sequence of operation in actual engineering system. In Equation (7), there exists second term $a_a \cdot b_b \cdot b_a$, because "*m*" is not quantitatively determined, and $a_a \cdot b_b \cdot b_a$ is not determined as a subset of $m \cdot a_a \cdot b_b$.

B.3. Determination of Arbitrary Set "m"

Now, consider the process of making loop operation, and obtain exact value of *m* in Equation (7).

At time *t1*, start the components A_b and B_a. A set $a_b(t1)$ is defined as component A_b is in operating state at time *t1*. Next at time *t2*, B_b is started, and inputs to A_a become $b_a(t2)b_b(t2)$ and $a_b(t2)$. But A_a is not started and there is no output from A_a. At time *t3*, start the component A_a. The outputs of A_a become $b_a(t3) \cdot b_b(t3) \cdot a_a(t3) + a_a(t3) \cdot a_b(t3)$, it is equal to "*a*" (output of A_a) and becomes to additional input to B_b as shown in Fig. 4.

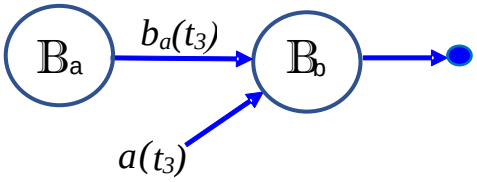

Fig. 4 Additional input "a" to B_b..

There is takeover phenomenon between $b_a(t_3)$ and $a(t_3)$, and output of B_b ("$b(t_3)$") becomes as follows [9].

$$b(t_3) = b_a(t_3) \cdot b_b(t_3) + b_a(\tau_3) \cdot b_b(t_3) \cdot a_a(t_3) \qquad (8)$$

Where τ_3 is a fixed value of time t_3 and $b_a(\tau_3)$ is also a fixed value, not a variable. Then, "*a*" (output of A_a) becomes,

$$a(t_3) = b(t_3) \cdot a_a(t_3) + a_b(t_3) \cdot a_a(t_3) = b_a(\tau_3) \cdot b_b(t_3) \cdot a_a(t_3) + b_a(t_3) \cdot b_b(t_3) \cdot a_a(t_3) + a_b(t_3) \cdot a_a(t_3) \qquad (9)$$

An arbitrary set "*m*" is determined as $b_a(\tau_3)$ compare to Eq. (7). As non-repairable model is assumed, $b_a(\tau_3)$ always include $b_a(t)$; $t>t_3$. Therefore, Eq. (9) finally becomes,

$$a(t_3) = b_a(\tau_3) \cdot b_b(t_3) \cdot a_a(t_3) + a_b(t_3) \cdot a_a(t_3) \qquad (10)$$

Equation (6) can be solved by Boolean arithmetic calculation with a consideration of takeover phenomenon.

B.4. Solution of the Simple Dependent Fault Trees

Now obtain the solution of fault tree shown in Fig. 1, from the Equation (10). The Equation (10) is converted into failure expression for *t>t3*.

$$A(t) = A_b(t)B_a(\tau_3) + A_b(t)B_b(t) + A_a(t) \qquad (11)$$

This exact solution is different from the one which is simply obtained by cut off the term *A* as shown by Equation (3) in section *A*. The factor $B_a(\tau_3)$ appears in the present starting process of physical system operation.

IV. Complex Fault Trees

In this chapter, the procedure presented in the previous section is applied to more general loop structured system, and confirm this procedure is generally applicable to fault tree with feedback loop [11].

Take up fault trees expressed in the set of Equation (12) to Equation (14).

$$A = A_a + A_b \cdot B + A_c \cdot C + A_{bc} \cdot B \cdot C \qquad (12)$$
$$B = B_b + B_a \cdot A + B_c \cdot C + B_{ac} \cdot A \cdot C \qquad (13)$$
$$C = C_c + C_b \cdot B + C_a \cdot A + C_{ab} \cdot A \cdot B \qquad (14)$$

Convert the Equation (12) to Equation (14) into success event expression.

$$a = a_{bc}a_a a_b a_c + a_a a_c b + a_a a_b c + a_a bc \qquad (15)$$
$$b = b_{ac}b_a b_b b_c + b_b b_a c + b_b b_c a + b_b ca \qquad (16)$$
$$c = c_{ab}c_a c_b c_c + c_c c_b a + c_c c_a b + c_c ab \qquad (17)$$

The set of Equations (15), (16) and (17) represent rather complicated but symmetrical loop structured system as shown in Fig. 5.

Then, consider a physical system model which corresponds to Fig. 5. By the same procedure shown in section III-B, top event *A* is obtained as Equation (18). This solution is deduced by a physical system model with some specific operational sequence. There is no unknown element in this equation.

$$TOP(A) = A_a(t) + A_b(t) \cdot B_b(t) + A_b(t) \cdot B_c(t) \cdot C_c(t) + A_c(t) \cdot C_c(t) + A_c(t) \cdot C_b(t) \cdot B_b(t) + A_{bc}(t) \cdot C_c(t) \cdot B_b(t) + A_{bc}(t) \cdot C_b(t) \cdot B_b(t)$$
$$+A_{bc}(t) \cdot B_c(t) \cdot C_c(t) + A_b(\tau_3) + A_c(\tau_4) + A_{bc}(\tau_3) + A_b(t) \cdot B_c(\tau_4) + A_c(t) \cdot B_c(\tau_4) + A_c(t) \cdot B_c(\tau_5) + A_{bc}(t) \cdot B_c(\tau_4) + A_{bc}(t) \cdot B_b(\tau_5) \quad (18)$$

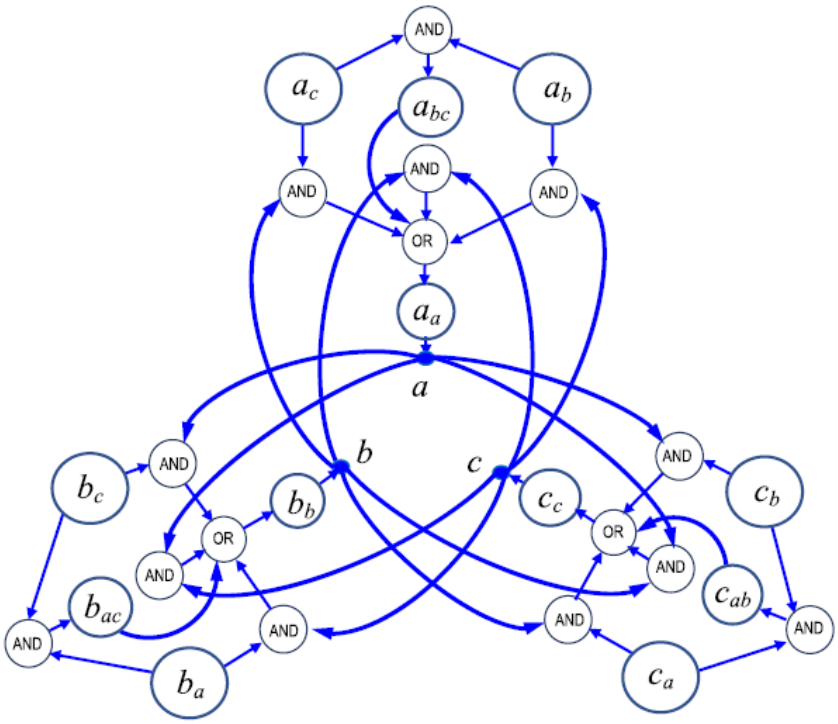

Fig. 5 Relations between success events in Equations (15), (16) and (17).

V. CONCLUSIONS

This paper presents a method to solve a Fault Tree with feedback loop in success event expressions.

Simple FTs (FTs with ordinary loop) and 3 non-linear interrelated FTs are taken up and converted into success event expressions. Corresponding system models, which satisfy Boolean equations in success event expressions, are deduced. And possible operating states are identified by considering these physical system models' structures and starting orders of component's operations.

It is shown that FT with feedback loops can be solvable without approximation. The implementation of this procedure to traditional FTA software tools or some other tools is a future work.

REFERENCES

[1] G.A. Coles, T.B. Powers, Breaking the Logic Loop to Complete the Probabilistic Risk Assessment, Proceedings of PSA'89, Pittsburgh, USA, pp. 1155-1160, 1989.

[2] J.E. Yang, S.H. Han, J.H. Park , Y.H. Jin, Analytic method to break logical loops automatically in PSA, *Reliability Engineering and System Safety*, Vol. 56, pp.101-105, 1997.

[3] J.K.Vaurio, A Recursive method for breaking complex logic loops in Boolean system models, *Reliability Engineering and System Safety*, Vol.92, pp.1473-1475, 2007.

[4] H.G. Lim, C.S. Jang, An analytic solution for a fault tree with circular logics in which the systems are linearly interrelated, *Reliability Engineering and System Safety*, Vol. 92, pp. 804-807, 2007.

[5] T. Matsuoka, An exact method for solving logical loops in reliability analysis, *Reliability Engineering and System Safety*, Vol.94, pp. 1282-1288, 2009.

[6] W.S. Jung, S.H. Han, J. Ha, A fast BDD algorithm for large coherent fault trees analysis, *Reliability Engineering and System Safety* Vol.83, pp. 369–374, 2004.

[7] Y. H. Chae, S. G. Kim, P. H. Seong. Reliability of the system with loops: Factor graph based approach. *Reliability Engineering and System Safety*, Vol.208, 2021, 107407

[8] T. Matsuoka, Generalized method for solving logical loops in reliability analysis, Proceedings of PSAM11, Helsinki, Finland, June 2012.

[9] T. Matsuoka, Dynamic Behavior of Nuclear Power Plant State under Severe Accident Conditions; Analysis by the GO-FLOW Methodology and the Consideration of Loop Structures. In: T. Aldemir, Editors. Advanced concepts in Nuclear Energy Risk Assessment and Management, World Scientific Publishing Co Pte Ltd, pp.427-476, 2018.

[10] T. Matsuoka, Method for solving logical loops in system reliability analysis. *Nuclear Safety and Simulation*; Vol.1, pp.328 – 339, 2010.

[11] T. Matsuoka, Procedure to solve mutually dependent Fault Trees (FT with loops), *Reliability Engineering and System Safety*, Vol.214, 2021, 107667.