Nuclear Engineering and Design 243 (2012) 336–340

Contents lists available at SciVerse ScienceDirect

Nuclear Engineering and Design

journal homepage: www.elsevier.com/locate/nucengdes

Top event probability evaluation of a fault tree having circular logics by using Monte Carlo method

Sang Hoon Han*, Ho-Gon Lim

Korea Atomic Energy Research Institute, 1045 Daedeokdaero, YuseongGu, Daejeon 305-343, Republic of Korea

ARTICLE INFO

Article history:
Received 26 May 2011
Received in revised form 7 November 2011
Accepted 22 November 2011

ABSTRACT

The current quantification method using minimal cut set generation in a PSA (Probabilistic Safety Assessment) uses several approximations for practical reasons. The Monte Carlo approach can be used as a supplementary means to verify the results of the minimal cut set approach. A new quantification algorithm using Monte Carlo sampling has been developed to handle a fault tree having circular logic. As a pilot application, two event trees of loss of off-site power and station blackout for a nuclear power plant have been quantified using the presented algorithm, which shows large conservatism in the conventional fault tree quantification method.

1. Introduction

The value of an event sequence in a PSA (Probabilistic Safety Assessment) is estimated conservatively due to approximations usually used in the typical quantification software using the minimal cut set approach. One is the delete-term operation to treat success logics, and the other is the rare event approximation or the minimal cut upper bound approach to calculate the value for the minimal cut sets (Epstein and Rauzy, 2005).

An event tree sequence modeled in a PSA can be expressed as a Boolean equation (or a fault tree), which consists of failure branches and success branches with an initiating event in its top as follows:

$$Seq = IE \cdot \Pi(BF_i) \cdot \Pi(\neg BS_j) \tag{1}$$

where IE is an initiating event, BF_i is a branch corresponding to the failure of a system/function, BS_j is a branch corresponding to the success, and ¬ represents logical negation. To quantify the frequency of an event scenario, the minimal cut sets are generated for the logic excluding success branches in most PSAs. The logic for success branches are approximated using a delete-term operation (Epstein and Rauzy, 2005) as below:

$$Seq' = IE \cdot \Pi(BF_i) \tag{2}$$

$$Seq'' = Seq' \Theta \Sigma(BF_j) \tag{3}$$

where Θ represents a delete-term operation, which deletes a cut set in Seq' if the cut set is a super set of any cut set in BF_j.

Also, the value for the sequence is approximated by the minimal cut upper bound or the rare event approximation. Thus, this approach may produce conservative results in most quantifications of a PSA model. In particular, when a PSA model has an independent event of high probability such as human error, it produces a large deviation from the exact value of quantification.

As an alternative of the conventional quantification method, the BDD method (Rauzy, 1993; Sinnamon and Andrews, 1997) has been researched as one of the approaches because it does not use any approximation in its quantification process. However, it is known that it cannot solve a large fault tree such as a PSA model for a nuclear power plant. This is due to the fact that the current computing power does not support the full BDD for large fault trees.

The Monte Carlo approach is applied in this article to evaluate the frequency of an event sequence without any approximation. The Monte Carlo approach has already been used to evaluate the top event probability of a large fault tree (Kumamoto, 1980; Kamat and Riley, 1975).

The minimal cut set approach is used in PSAs for the practical reason. Sometimes a supplementary means is required to verify the quantification value, especially for a fault tree having negates (success sequences of an event tree) or a fault tree having high value events. Software like DPC (Riley, 2009) is available to evaluate the exact value of a fault tree, but it does not handle a fault tree having circular logic. Thus, a new algorithm based on the Monte Carlo approach is developed to verify results of a PSA having circular logic in this article.

The algorithm of Monte Carlo sampling to evaluate the top event probability of a fault tree is described in Section 2. The newly developed algorithm to treat circular logic is presented in Section 3. The pilot application for event sequences of a PSA is presented in Section 4.

* Corresponding author. Tel.: +82 42 868 8921.
E-mail addresses: shhan2@kaeri.re.kr (S.H. Han), hglim@kaeri.re.kr (H.G. Lim).

0029-5493/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.nucengdes.2011.11.029

2. Algorithm for evaluating top event probability of a fault tree not having circular logic

In this section, a main algorithm for Monte Carlo sampling is presented to evaluate the top event probability of a fault tree that does not have circular logic. The algorithm consists of determining the state of each event randomly and calculating the state of the top event.

The top event of a fault tree can be expressed as a function of a basic event as follows:

$$T = f(E) \tag{4}$$

where T is a top event of a fault tree and E is a vector of a basic event as follows:

$$E = (E_1,\ E_2, \ldots, E_n) \tag{5}$$

where E_i is an i-th basic event of a fault tree. The state of all basic events and its occurrence probability are defined below

$$S = (S_1, S_2, \ldots, S_n) \tag{6}$$

where S_i is the state of the i-th event that can be True or False.

P_i: probability of E_i

In the Monte Carlo method, the probability of T is stochastically determined by sampling the state of basic events based on their occurrence probability, and propagating their states until the state of the top event is determined. The state vector S is determined by random sampling. By inserting the state vector S into Eq. (4), one can find the state of the top event. By repeating the same process sufficiently, one can obtain asymptotic probability of the top event. This algorithm can be summarized as follows:

```
Function EvaluateTopEventProbability ($T$)
  //for number of samples $n$
  For number of samples $n$
    //determine the state of events
    For each $E_i$ which is a basic event
      //a random number $r$ between 0 and 1
      $r$ = rand()
      //State of $E_i$
      if $r \le Pi$ then $S_i$ = True else $S_i$ = False
    //Calculate the state of top event
    $S_T$ = GetState (T)//$f(S)$
    if $S_T \equiv$ True then $m = m + 1$
  //Evaluate the top event probability $P_T$
  $P_T = m/n$
  Return $P_T$
```

The above algorithm determines the state of a basic event using random numbers for every sample. If a reduction of computation time and variance is required, the dagger sampling technique (Kumamoto, 1980) can be used.

The propagation of the state of a basic event can be performed recursively in the structure of a fault tree. Usually, a fault tree contains a large number of intermediate gates to handle the relation among basic events. To determine the state of the top event, the state of the intermediate gate should be determined. If the fault tree does not have circular logic, the algorithm is as shown below:

```
Function GetState ($g$)
  //return $S_g$ for basic event
  if $g$ is a basic event then
    return $S_g$
  //for Gate
  //if already processed
  if $S_g$ is already determined then
    return $S_g$
  //calculate the state for each input
  For each $C_i \in g$
    //recursively for each input
    $S_i$ = GetState ($C_i$)
  //Determine the State of $g$
  $S_g$ = DetermineState ($g$)
  Return $S_g$
```

The state of a gate S_g can be calculated based on the states of its inputs. If the states of all inputs for an OR gate are False S_g becomes False, otherwise it becomes True. If the states of all inputs for an AND gate are True, S_g becomes True, otherwise it becomes False. The state for NOT gate becomes a negation of the state of the input event. The algorithm is as shown below:

```
Function DetermineState ($g$)
  if $g$ is a OR Gate then
    u = False
    For each $C_i \in g$
      $u = u \cup S_i$
  else if $g$ is a AND Gate then
    u = True
    For each $C_i \in g$
      $u = u \cap S_i$
  else if $g$ is a NOT Gate then
    For $C_i \in g$
      u = not $S_i$
  $S_g$ = u
  return $S_g$
```

3. Evaluating top event probability of a fault tree having circular logic

3.1. Review of a method to break circular logics in a fault tree

Circular logic (or logical loops) can be found in a fault tree for a PSA of a nuclear power plant because some systems support each other. For example, a component cooling water system provides cooling water to a diesel generator, and in turn, the diesel generator provides electrical power to the component cooling water system.

Yang has developed an algorithm to break the circular logic in a fault tree automatically (Yang et al., 1997). The process to derive the algorithm is a little complex. However, the key idea of the algorithm is that the terms that cause the circular logic are identified during the expansion and are deleted.

Let us show an example of how to break the circular logic for the following fault tree given by Yang (Fig. 1):

Fig. 2 represents how to break the circular logic for top event A in Yang's algorithm, which consists of copying, renaming, and modifying gates related to the circular logic. A gate making a circular logic is deleted while expanding a fault tree. Gates marked with a bold X make circular logic and should be deleted (treated as False).

The process produces a logically equivalent fault tree not having circular logic. Please note that B′ is different than B″, both of which are derived from B.

Yang's algorithm is implemented in several cut set generation software such as KIRAP (Yang et al., 1997) or FTREX (Jung et al., 2005). Thus, some PSA software such as AIMS-PSA (Han et al., 2010) or KIRAP permit the circular logic in a fault tree.

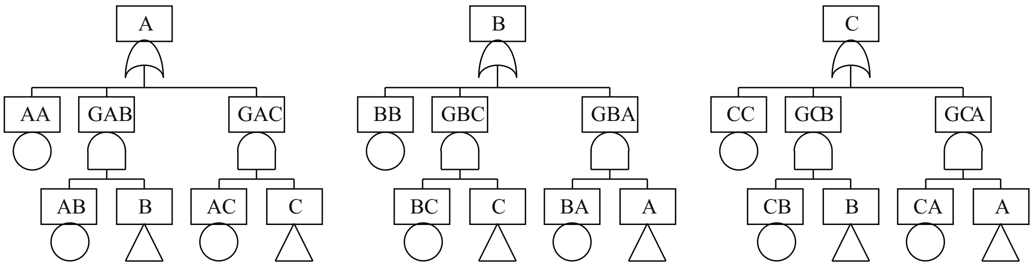

Fig. 1. An example of fault tree having circular logic.

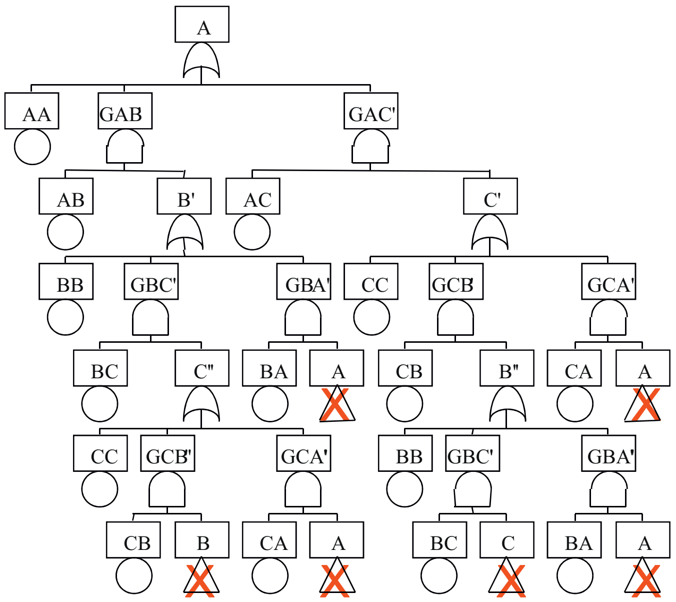

Fig. 2. An example to break circular logic.

3.2. *Recursive algorithm for evaluating top event probability of a fault tree having circular logic*

The procedure developed by Yang consists of copying, renaming, and modifying gates so as to convert a fault tree having circular logic into a logically equivalent fault tree that has no circular logic, as shown in Fig. 2. This process essentially produces a larger fault tree than the original fault tree, and also requires complex programming. This process cannot be avoided if the minimal cut sets are required.

The Monte Carlo approach requires a calculation of the state of a fault tree, not the minimal cut sets of the fault tree. It is possible to directly calculate the state of a fault tree without producing a logically equivalent fault tree.

Let us review Yang's idea, which is to delete gates that make circular logic while expanding the fault tree. Suppose a gate X that makes a circular logic. Then, using a Boolean expression, it can be expressed as below:

$$X = A + B \cdot X \tag{7}$$

where A is a Boolean logic not related to the circular logic, and B is a Boolean logic related to the circular logic. To solve X, X in the right-hand side is replaced with $A+B{\cdot}X$. The process will be repeated infinitely.

$$X = A + B \cdot X = A + B \cdot (A + B \cdot X) = A + B \cdot (A + B \cdot (A + B \cdot X)) \ldots \tag{8}$$

The term X in the right-hand side is treated as False if the same gate appears during the expansion, which produces the following:

$$X = A + B \cdot X = A \tag{9}$$

This idea can be simply implemented when calculating the state of a gate that makes a circular logic in the Monte Carlo approach. The new algorithm replaces the GetState function developed for a fault tree that does not have circular logic in Section 2.

A variable D is introduced to check the circular logic.

$$D = (D_1, D_2, \ldots, D_n) \tag{10}$$

D_i is a temporary variable used to check whether it makes a circular logic while traversing the fault tree recursively. If the value of D_i is larger than 1, the circular logic is encountered, and False is returned. Before calculating the state of a top event, all D_i should be initialized as 0.

```
Function GetState ($g$)
    //return $S_g$ for basic event
    if $g$ is a basic event then
        return $S_g$
    //For Gate
    //Increase Depth
    $D_g = D_g + 1$
    //if $g$ encounters itself
    if $D_g > 1$
        //delete the logic
        $S_g$ = False
    else
        //For each input
        For each $C_i \in g$
            //calculate the state recursively
            $S_i$ = GetState ($C_i$)
        //Determine the State of $g$
        $S_g$ = DetermineState ($g$)
    //Decrease depth
    $D_g = D_g - 1$
    //return result
    Return $S_g$
```

4. Application to PSA

The results from the minimal cut set approach and the Monte Carlo approach are compared for 2 event trees of a PSA model. The PSA model uses the small event tree and large fault tree approach, and has circular logic in its fault trees. The PSA model includes 3371 gates and 2832 basic events. The event trees for comparison are shown in Fig. 3. The 27-th sequence of the first event tree is transferred to the initiating event of the second event tree.

The event sequences, whose probabilities are larger than 10^{-7}, are selected and tested using the Monte Carlo approach. The calculation is done for the failure sequences as well as success sequences because this study was initiated to select sequences for investigating the uncertainty due to the thermal hydraulic calculation.

Each branch in event trees is represented by a fault tree or a basic event. Each event sequence is represented by a fault tree whose type is an AND gate of success and failure branches, as described in

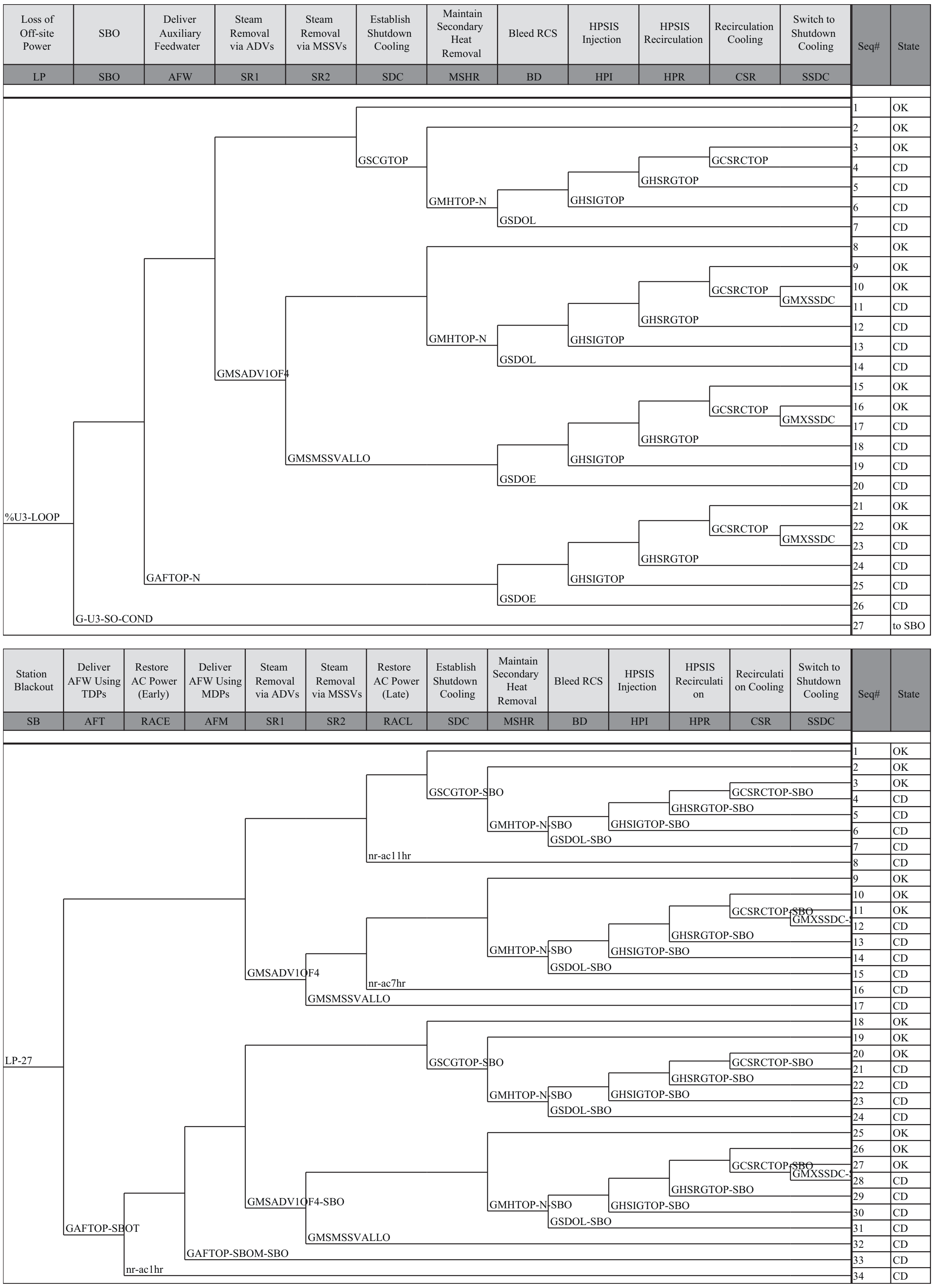

Fig. 3. Event trees for comparison.

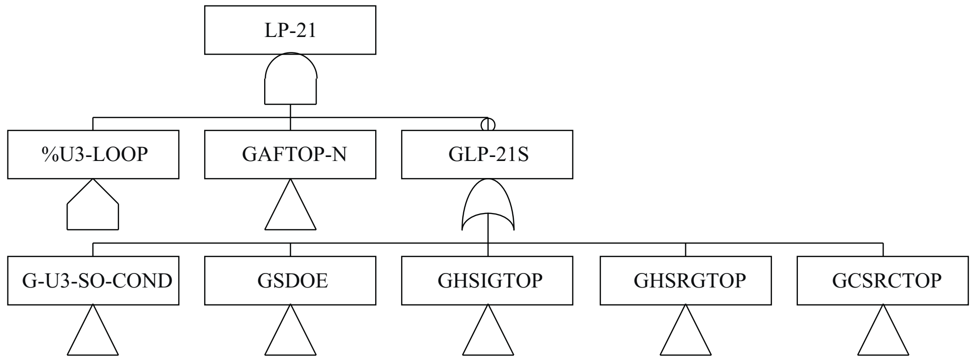

Fig. 4. A fault tree for an event sequence.

Table 1
Results from both approaches.

Sequence	State	CS[a]	MC[b]	Deviation[c]	Difference
LP-01	ok	1.00[d]	9.89e−1	ε	1.1%
LP-02	ok	1.02e−2	9.18e−3	5.55e−6	*11.5%*
LP-03	ok	2.91e−5	2.53e−5	3.77e−7	*15.1%*
LP-04	cd	3.20e−6	2.65e−6	2.06e−7	*20.8%*
LP-06	cd	1.10e−7	9.80e−8	3.29e−8	12.6%
LP-07	cd	1.88e−5	1.64e−5	3.38e−7	*14.7%*
LP-08	ok	1.20e−3	1.19e−3	6.36e−7	1.3%
LP-09	ok	6.34e−6	6.29e−6	4.12e−7	0.8%
LP-14	cd	5.33e−7	5.32e−7	9.09e−8	0.1%
LP-21	ok	2.40e−5	1.71e−5	2.45e−7	*40.1%*
LP-22	ok	1.35e−7	9.40e−8	1.51e−8	*43.9%*
LP-26	cd	2.56e−5	2.26e−5	4.21e−7	*13.6%*
SB-01	ok	8.19e−4[e]	7.50e−4	1.67e−6	9.2%
SB-02	ok	3.80e−6	3.46e−6	1.35e−7	9.9%
SB-08	cd	3.19e−5	3.07e−5	4.19e−7	4.1%
SB-09	ok	2.03e−5	1.80e−5	6.48e−7	*13.1%*
SB-10	ok	1.15e−7	8.70e−8	3.40e−8	32.0%
SB-16	cd	2.03e−6	1.97e−6	1.11e−7	3.1%
SB-18	ok	1.43e−5	5.18e−6	2.03e−7	*175.2%*
SB-34	cd	8.84e−6	8.58e−6	2.28e−7	3.0%

[a] Results from minimal cut set approach, cutoff = 10^{-14}.
[b] Results from Monte Carlo approach, the number of samples = 10^9.
[c] Standard deviation from Monte Carlo approach which represents the uncertainty.
[d] The frequency of the initiating event is treated as 1.
[e] The initiating event is the transferred sequence from the 27-th sequence of the first event tree.

Eq. (1), which is the typical approach called the fault tree linking approach. For example, the fault tree for the 21-th sequence of the first event tree, LP-21, is represented in Fig. 4, where the GLP-21S is corresponding to the success branches.

The calculation time for Monte Carlo simulation takes about 3 days with 10^9 of the number of samples. In practical, 10^7–10^8 of the number of samples will be enough for important sequences whose probabilities are large.

The results show that the minimal cut set approach produces conservative values in most cases. The difference is marked in italics for each sequence if it is larger than 10% and beyond 95% of the expected value considering uncertainty. The minimal cut set approach produces results 40% larger than the Monte Carlo approach for some sequences. The main reason for differences is presumed because the delete-term approximation is used to treat the success branches that have larger probability than 0.1 (Table 1).

5. Conclusions

Since the typical PSA uses several approximations to obtain minimal cut sets and to quantify the occurrence frequency of a fault tree, its estimation may have uncertainty in its values. For example, the success scenario of a PSA model has larger uncertainty because most parts of this scenario consist of successes of system/function thereby the delete-term approximation has larger effect on the quantification. The minimal cut upper bound approach may also produce a large uncertainty if a fault tree includes events whose probabilities is large.

An algorithm was developed to evaluate the top event probability for a fault tree that has circular logic using the Monte Carlo approach. The Monte Carlo approach is simple and easy to implement in a program. It provides a good estimation of a real value if the number of samples is sufficient.

This Monte Carlo approach requires a lot of computation time, and does not provide information for event importance measure. However, it can be used as a supplementary means to verify the results of the minimal cut set approach for PSAs which includes circular logic in fault trees.

Acknowledgement

This work was performed under the Long-term Nuclear R&D Program sponsored by the Ministry of Education, Science and Technology.

References

Epstein, S., Rauzy, A., 2005. Can we trust PRA? Reliability Engineering and System Safety 88, 195–205.
Han, S.H., et al., 2010. Improved features in a PSA software AIMS-PSA. In: Transactions of the Korean Nuclear Society Spring Meeting, Korea.
Jung, W.S., et al., 2005. Development of an analytical method to break logical loops at the system level. Reliability Engineering and System Safety 90, 37–44.
Kamat, S.J., Riley, M.W., 1975. Determination of reliability using event-based Monte Carlo simulation. IEEE Transactions on Reliability R-24 (1), 73–75.
Kumamoto, H., 1980. Dagger-sampling Monte Carlo for system unavailability evaluation. IEEE Transactions on Reliability R-29 (2), 122–125.
Rauzy, A., 1993. New algorithms for fault trees analysis. Reliability Engineering and System Safety 40, 203–211.
Riley, J., 2009. DPC User Manual Version 4.0. EPRI.
Sinnamon, R.M., Andrews, J.D., 1997. New approaches to evaluating fault trees. Reliability Engineering and System Safety 58, 89–96.
Yang, J.E., et al., 1997. Analytic method to break logical loops automatically in PSA. Reliability Engineering and System Safety 56, 101–105.