Progress in Nuclear Energy 85 (2015) 192–199

Contents lists available at ScienceDirect

Progress in Nuclear Energy

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

BDD algorithms based on modularization for fault tree analysis

Yunli Deng, He Wang*, Biao Guo

Fundamental Science on Nuclear Safety and Simulation Technology Laboratory, College of Nuclear Science and Technology, Harbin Engineering University, 145-1 Nantong Street, Nangang District, Heilongjiang, Harbin, 150001, PR China

ARTICLE INFO

Article history:
Received 4 March 2015
Received in revised form
11 May 2015
Accepted 16 June 2015
Available online xxx

Keywords:
Binary decision diagram
Modularization
Fault tree analysis
Probabilistic Safety Assessment
Nuclear safety

ABSTRACT

This paper explains a Binary Decision Diagram (BDD) algorithm based on modularization to solve a Fault Tree (FT) or Event Tree (ET) in Probabilistic Safety Assessment (PSA) of the nuclear power plant. Fault Tree Analysis (FTA) is a non-deterministic polynomial-time hard problem, both the complexity of the calculation and the large size of real models exponentially increase along with the number of variables, as well as the BDD structure which is transferred from FT model. When solving a large FT in BDD algorithm, two difficulties will be met: the memory is not enough to store large BDD structure and the process generates too much Minimal Cut Sets (MCSs) that is time consuming.

In order to overcome these difficulties, this paper presents new insights on integrating truncation means and BDD based on modularization technique. The results indicate that the algorithm is efficient to solve the FT or ET with less memory consumption and in shorter time. This new idea can be applied to Risk Monitor or On-line Risk Monitor of nuclear power plant, which makes the computing process more efficient and fast.

1. Introduction

The Fault Tree (FT)/Event Tree (ET) methodology is widely used in the nuclear industry to obtain the response models for Probabilistic Safety Assessment (PSA). Almost, all probabilistic analyses carried out in nuclear safety studies rely on these models (Ibáñez et al., 2008). Many computer packages are available, which are based on conventional kinetic tree theory methods (Vesely, 1970). When dealing with large FTs, the limitations of the technique in terms of accuracy of the solutions and the efficiency of the computing process become apparent. Over recent years, the binary decision diagram (BDD) method has been developed to solve FTs and overcome the disadvantages of the conventional fault tree analysis (FTA) approach.

In BDD method, the FT is converted to a BDD structure, which represents the Boolean logic expression of the particular system failure mode. But FT model and BDD structure exponentially increase along with the number of variables, and the size of BDD structure is very sensitive to variable ordering as described by Vesely, 1970 and Awinash et al., 2010 (Vesely, 1970; Kumar et al, Dec 2010). Therefore, when solving a large FT in BDD algorithm, the high memory consumption is a limitation. In addition, a large FT includes large number of Minimal Cut Sets (MCSs) which may be thousands or more. If MCSs are used to identify weaknesses and strengths of the system, it is often hard to identify these when there are thousands of MCSs. Furthermore, as the size of BDD structure and the number of MCS increase, the computation time increases.

In order to overcome these difficulties, this paper presents new insights on integrating truncation means and BDD based on modularization technique. Compared with traditional BDD algorithm, new algorithm will reduce the memory consumption and computation time. Usually, an FT or even a large FT is converted to a whole BDD structure firstly following the subsuming and traversing with truncation. The BDD structure may be too large to be stored in computer memory during this process. However, finding the modules is the first step in the new algorithm, then it begins with traversing the first module, if the BDD structure has not been formed before this, the module needs to be converted to BDD structure, it is same as the first module when traversing other modules. Of course, truncation method is applied in the traversing process, many modules have been cut off before they are converted to BDD structures if there is a proper truncation probability.

In FT modularization method, the repeated structures are saved only once into the memory and it reduces the large size BDD structure into smaller ones. Moreover, the combination of modularization method with probability truncation means cuts off many

* Corresponding author.
E-mail address: wangheboy@hotmail.com (H. Wang).

http://dx.doi.org/10.1016/j.pnucene.2015.06.019
0149-1970/

modules. It means that many BDD structures do not need to be formed to store into computer memory. These will significantly decrease the memory consumption in computation process. Meanwhile, compared with traditional BDD algorithm, it can get favorite outcomes including accurate failure probability of top event and fewer MCSs in a shorter time.

This paper combines truncation means with BDD based on modularization algorithm, which will have more advantages when the idea is applied to Risk Monitor or On-line Risk Monitor of the nuclear power plant. The key to the Risk Monitor and On-line Risk Monitor system is the rapid calculations according to PSA model changes. The process of recalculating the large PSA model is very time consuming (Kim and Han, 1998), however, the calculations of only those modules where the changes take place will take a very short time. Therefore, if there is some technique to identify those modules where the model has changed, the calculation with the new algorithm described in this work will be much more efficient.

2. Binary decision diagrams

A BDD is a directed acyclic graph that depicts the Shannon decomposition and a data structure that is used to represent a Boolean function (Bryant, 1986) as shown in Fig. 1(a). All paths through the BDD start at the root vertex and terminate in one of two states that labeled 0 and 1, encoding two states of system (failure and success). Additionally, a variable node has two outgoing edges to collect to other variable nodes or terminal nodes, called 0-branch and 1-branch, which also represent the variable node being on failure or success state.

Rauzy proposed an algebraic framework to compute MCSs of FT based on Shannon decomposition (Antoine, 1993), The general Shannon decomposition is succinctly defined in terms of ternary If-Then-Else (ITE) connectives as

$$f = ite(x, f_1, f_0) = xf_1 + \bar{x}f_0 \tag{1}$$

Where x is one of variables and the functions f_1 and f_0 are Boolean functions evaluated at $x = 1$ and $x = 0$.

In the reference (Antoine, 1993), Rauzy put forward a set of ITE connectives rules in which FT can be converted to a BDD structure. Basic events of FT are the variable nodes in BDD, an independent node collects to 1 terminate and 0 terminate with 1-branch and 0-branch as shown in Fig. 1(b). Basic events are collected by “and” and “or” gates in a FT (coherent), Rauzy's rules are explaining the ‘GATE’ operation between two ITE connectives. If x and y are two variables with $x<y$, then the following equalities are.

For “and” gate operation:

$$ite(x, G_1, G_2)ite(x, H_1, H_2) = ite(x, G_1H_1, G_2H_2) \tag{2}$$

$$ite(x, G_1, G_2)ite(y, H_1, H_2) = ite(x, G_1h, G_2h) \tag{3}$$

For “or” gate operation:

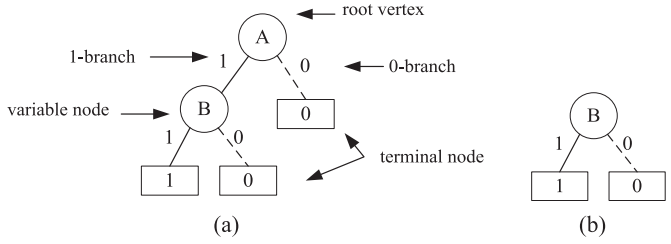

Fig. 1. Interpretation of BDD structures.

$$ite(x, G_1, G_2) + ite(x, H_1, H_2) = ite(x, (G_1 + H_1), (G_2 + H_2)) \tag{4}$$

$$ite(x, G_1, G_2) + ite(y, H_1, H_2) = ite(x, (G_1 + h), (G_2 + h)) \tag{5}$$

where $h = ite(y,H_1,H_2)$

A BDD structure can be constructed using Equations (2)–(5) for any fault tree, once a total ordering of variables is selected (see an example in Fig. 2). It can generate MCSs and calculate the top probability directly by traversing the BDD structure of a FT. As illustrated in Fig. 2(b), the BDD structure has$\{a, c\}, \{a, \bar{c}, d\}, \{a, \bar{c}, \bar{d}, e\}$ and $\{a,b\}$ four paths with 1 terminate, so {ac,ad,ae,ab} are the MCSs of the FT. The top event probability can be figured out in Equation (6) without the need to apply approximations (Remenyte and Andrews, 2008).

$$P_T = P_aP_c + P_a(1 - P_c)P_d + P_a(1 - P_c)(1 - P_d)P_e + P_aP_b \tag{6}$$

BDD algorithm is an efficient method to perform FTA (Remenyte and Andrews, 2008, 2006), however, the size of BDD structure exponentially increasing according to the number of variables, it has expensive memory consumption. When handing with a large FT, it is impossible to convert the FT to BDD with personal computer, even if adopting the memory management for BDDs referring in (Antoine, 1993). The modularization method for FTs can disassemble a large FT into small independent subtrees whose BDDs can be constructed in less memory consumption. Therefore, the modularization method makes it possible that the large FTs especially the ones of nuclear power plant can be solved.

3. Modularization method

The modularization is a method to find independent modules of an FT (Dutuit and Rauzy, 1996). A module is defined as: a subtree composed of at least two events which have no inputs from the rest of the tree and no outputs to the rest except from its output events (Kohda, 1989), in simple words, a subtree contains no basic events that appear elsewhere in the FT. The advantage of identifying these modules is that each one can be analyzed separately from the rest of the tree (Reay and Andrews, 2002).

A depth-first left-right which is a linear-time algorithm is employed to identify modules of an FT. There are two depth-first left-right traversals of the FT, the FT's structure is recorded after the first traversal, and the second traversal is to get the every visit time of each gate and basic event in FT. Starting at top event and progressing the FT in a depth-first left-right manner, it should be noted that each gate is visited at least twice: once on the way down the tree and again on the way back up the tree, and there is a rule that if a gate has been visited, it can be visited again, but its children events or gates should not be visited.

It is a module, if the subtree is satisfied with the condition that visit times of any children must be between the first visit time and second visit time of its top event. For each gate or event, it needs to record the first visit time, second visit time and last visit time, if a basic event just be visited only once, its second visit time and last visit time equals to the first visit time, and if a basic or gate just be visited twice, the second visit time is the last visit time. To demonstrate this, refer to the FT mentioned in Fig. 3, the gates and their visit times are given in Table 1 and the basic events are given in Table 2.

The maximum visit time and minimum visit time of the children for a subtree equal to *Max* and *Min*, if the interval[*Min*, *Max*] belongs to $[T_f,T_s]$, the subtree is a module. The modularization results of the example FT are given in Table 3.